Diplomarbeit

Elajah Ngankepeh

Hardware and Software for Position Determination and Visualization for an Indoor Navigation System

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Elajah Ngankepeh

Hardware and Software for Position Determination and Visualization for an Indoor Navigation System

Diplomarbeit eingereicht im Rahmen der Diplomprüfung
im Studiengang Informations- und Elektrotechnik
Studienrichtung Automatisierungstechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Ing. Karl-Ragmar Riemschneider
Zweitgutachter : Prof. Dr. Ing. Henry Reetmeyer

Abgegeben am 08. Juli 2008

Elajah Ngankepeh

Thema der Diplomarbeit

Hard- und Software für die Positionsberechnung und Anzeige für ein Indoor Navigationssystem

Stichworte

LPS, LED Anzeigesystem, Mikrokontroller Programmierung, MSP430, LED Steuerung, MAX7221,USART ,SPI

Kurzzusammenfassung

Die Position eines Gegenstandes, zum Beispiel ein Roboter ist zu bestimmen und in einer Anschaulicherweise darzustellen. Die Laufzeitunterschiede werden von einem parallelen Prozeß empfangen, das diese Laufzeitunterschiede durch Korrelation anhand Schalls berechnet. Da es keine Synchronisierung zwischen Absender und Detektor gibt, sind nur Zeitunterschiede direkt ermittelbar und nicht die absoluten Laufzeiten. Ein 16x32 LED- Matrix- Anzeigesystem wurde entwickelt. Die Software wird in der C-Sprache und der Compiler GNU-Eclipse entwickelt. Der Hardware-Entwurf ist unter Verwendung von Eagle realisiert.

Elajah Ngankepeh

Title of the paper

Hardware and Software for Position Determination and Visualization for an Indoor Navigation System

Keywords

LPS, LED Display system, microcontroller programming, MSP430, MAX7221, LED control, USART, SPI

Abstract

In this project, the position of an object for example a robot indoor is determined on the basis of time differences received from a parallel process. This process calculates the time differences through correlation based on sound. This sound is detected at different time, proportional to their distance to the source. Since there is no synchronization between sender and detector, only differences between the elapsed time are directly obtainable. A display is developed which serves the purpose of demonstrating the location of the object. The software is developed using the C language and the compiler GNU Eclipse. The hardware design has been realised using Eagle.

Acknowledgement

I would like to thank all those who participated in making this piece of work a reality. Special thanks go to my examiner Prof. Dr.-lng. Karl-Ragmar Riemschneider, who had been very available in helping and also for suggesting the thesis. I am equally thankful to my second examiner , Prof. Dr. Henry Reetmeyer who took the pains to go through the work and assisted actively in modification. Many thanks equally go to Mr. Jörg Pflüger and Mr. Wolff Gehard , who actively supported me during this thesis. I will like to dedicate this piece of work to my late dad: Moh Tangongho Abraham, not forgetting my mum, my brother Moh Sylvester Tangongho who made sure I attended the best school available. I also thank my sister Angela Ngamwe who was so caring and full of assistance. Very special thanks go to Ngoun Abiba who assisted me at all levels.
I will also like to cease this opportunity to thank all my class mates who participated passively as well as actively during this thesis.

CONTENTS

List of Figures

List of tables

Chapter 1: - Introduction

Science is a system of accumulating reliable knowledge. Broadly speaking, the process of sciences begins with observations, which are developed into hypothesis, tested by proof or experimentation, yielding results that can be described in a paper (scientific paper for example) which is published after a thorough reviewing . Each new contribution mounts on a bed of existing concepts that are known and trusted. New research could be wrong or misguided, but the process of referring eliminates work of poor quality. Determining the local position of an object in the process: Local Positioning System is proceeded from the principles of General Positioning System GPS. These principles involve sending signals, receiving these signals, carrying out correlation, determining the time difference between sender and receiver and finally processing the data to locate the position of the object as well as the rate of change of position; velocity. A major task in location-aware programming is the determination of physical location. Researches have created numerous location-sensing systems that differ in accuracy, coverage, frequency of location updates, and cost of installation and maintenance.

This piece of work involves the determination of the position of an object in a limited area. The limit is about 10 meters. This limit is based on the ability of the receivers to clearly detect the sound emitted. After the sound is detected, correlation takes place in a parallel project. The measurement of the time-of-flight (TOF) of a sonic signal propagating from an emitter to a receiver gives an indication of their relative distance or range. The run time differences are then calculated by multiplying the maximum correlation position with the period at which the sound is being emitted. The position at which maximum correlation occurs is received through Universal Asynchronous Receiver and Transmitter (UART) protocol communication. The MSP430 is the microcontroller used here as it has many advantages related to the ultra- low power ability for stand- alone systems. The received run time differences are then incorporated into the geometry calculation involving the speed of sound in air.

The most important factor influencing the speed of sound in air is temperature. This factor is considered in order to improve on the results as far as the accuracy is concerned. The temperature of air at the moment of carrying out the experiment is obtained using the MSP430F169. It has a diode, whose voltage fall is linearly related to the temperature of the environment. The measurement gives an accuracy of up to 1°C after calibration. This accuracy is sufficient to obtain better results in determining the position of the object. The obtained temperature is displayed on the Liquid Crystal Display (LCD) which is available on the MSP430F169 Starter Kit. This LCD can allow 32 characters to be displayed at the same time. Addressing the LCD is possible with a nibble (4 bits) by nibble or a byte (8 bits) by byte transfer. The positions of the four sensors (microphones) are fixed and given in using the three buttons available on the MSP430. The values are seen at the Liquid Crystal Display (LCD) during input. Values beyond the range are not accepted and set to either the maximum value or the minimum value depending on the extreme at which the user is exceeding. In the geometry calculation, it is first considered that the object whose position is to be determined, is positioned somewhere in the limited area. At the end of the calculation, the coordinates obtained are those signifying the difference between the real position and the estimated position. This means the real position of the object in question is then obtained by adding the estimated position to the calculated difference. This is clearly illustrated in **chapter two**.
The X and Y coordinates are to be represented on a Light Emitting Diode (LED) display board with 32x16 LEDs matrix. These coordinates are represented as a point. The Z-axis is not represented on the LED matrix board as it is two dimensional, though could be displayed

on the LCD every time new values are obtained. To be able to control the matrix boards, the chip MAX7221 from the company MAXIM was used. These chips are compact, serial input/output common-cathode display drivers that interface microprocessors (µPs) to 7-segment numeric LED displays of up to 8 digits, bar-graph displays, or 64 single LEDs. Included on-chip are a BCD code-B decoder, multiplex scan circuitry, segment and digit drivers, and an 8x8 static RAM that stores each digit. Only one external resistor is required to set the segment current for all LEDs. It is compatible with SPI, Queued Serial Peripheral Interface (QSPI) and MICROWIRE, and has slew-rate-limited segment drivers to reduce Electromagnetic Interference (EMI)

Controlling the LEDs from the MSP430 is through the Universal Synchronous and Asynchronous communication (USART) as Serial Peripheral Interface (SPI) latching the data out using a clock of the MSP430.

A Printable Circuit Board (PCB) has been designed in Eagle to facilitate connection. The electric circuit board was double sided and compact to suit the size of the LED matrix board and its pins.

1.1- Task Overview

The thesis involves developing software and hardware for visualization in an indoor navigation system using sound as the signal. Programming is in C with the compiler MSPGCC in Eclipse GNU. The microcontroller MSP430F169 Starter Kit is to be employed for the numerous communications involved. Some of these are UART and SPI. The UART communication serves in the reception of maximum correlation positions from a parallel project. These positions are to be processed into run time differences which are further incorporated in geometry calculation and the position of the object obtained.

The PCB design is to be realized using Eagle Version 4.11. This Design is to be used in the visualization module of the hardware, making the entire hardware less cumbersome. This module consisting of 512 LEDs and on 8 LED matrix Displays. The location of the object is to be represented on this display, and as a point. The 512 LEDs represent the bounded area of experiment or room and the representation is such that the LED surface could be mapped to the area being covered by the 4 sensors.

Some major factors like the temperature, reflection of sound are to be taken into consideration since the speed of sound in air is influenced by the temperature and sound is equally reflected by most objects.

1.2- Organization of the Thesis

Obtaining and processing the run time differences for four sensors, which have been obtained through correlation, are most of the theory behind this work, apart from the behaviour and emission of sound in air. The position (Cartesian coordinate) of an object could be determined in a range of about 10 meters. The good thing about it is that it could be applied in an open air likewise indoors. This special characteristic assists to complete certain general positioning systems that mainly work outdoor. The next and important point is the methodology to obtain the time differences, process them and finally represent the position of the object (for example a robot) in question, at any given time on the display. The low power mixed signal controller MSP430F169 has been of great use as it is the central controller for the entire project. It is used in obtaining the run time differences, lighting up specific Light Emitting Diodes (LED) corresponding to the position of an object/robot and also in determining the temperature of air at the moment of determining the position. This project is a sub project of a larger project. The run time differences and sound radiation are from other sub projects.

The first chapter consists of the general introduction of the project, including the structure of the entire project.

In **chapter two**, the general theory behind the project; processing run time differences of 4 sensors and displaying the position of the object on a 512 LED matrix board will be explained. A comparison or rather evolution from the theory of GPS to the development of LPS (Local Positioning System) is step by step explained here. It is as well elaborated, how the inverse of a 4x4 matrix is obtained. This is important to ease the coding in the C programming language. The entire code for this work is written in C using the GNU compiler in Eclipse. Eclipse version 1.3 was used.

The **third chapter** describes the mixed signal microcontroller MSP430 F169 Starter Kit and its peripheries. The general characteristics are first examined, followed by the ports of the board, then the universal synchronous and asynchronous transmission ability (USART). The asynchronous communication is outlined singly, followed by the synchronous communication (SPI: Serial Periphery Interface). This board has a USART0 (Universal Synchronous and Asynchronous Receiver and Transmitter 0) and a USART1 (Universal Synchronous and Asynchronous Receiver and Transmitter 1). These helped very much, as there was the necessity for a second periphery in order to carry out all the necessary communications. The first communication, being the continuous reception of run time differences when the object displaces it self or is displaced and the next is the communication (SPI) with the MAX7221 chips incorporated in the LED module.
The ability of the analogue-digital converter of the board aided in converting values obtained as voltage difference across a diode into temperature. There is an analogue-digital converter with 10 bits resolution (ACD10) as well as a 12 bits resolution: ADC12. The ADC12 was used in order to have a better resolution and as such a better accuracy.
A liquid crystal display (LCD) is also available on the board and the detail exploitation is outlined later in **chapter three**.

The development of the employed experimental board is explained in **chapter four**, not forgetting the tool employed: Eagle. The general structure and functionality of the LED matrix board is as well outlined here. MAX7221, which is the chip used in this project, has a few properties that make it convenient for controlling the LEDs. These properties are thoroughly treated in the **third chapter**.

The software part of the project is explained in **chapter five**. The programming is modular, with every little section separated to make the layout clear and permit easy detection and accessibility of errors in case the project is incorporated into a further project. There are many modules; one is calculating the inverse of a 4x4 matrix. A second module determines the temperature of air during the experiment. A third module permits the synchronous communication of the MSP430169 and the 8 LED matrix boards. A fourth module realises the asynchronous communication between the board and the source of the run time differences. These and other unlisted modules are discussed thoroughly in **chapter five**.

In **chapter six**, some special techniques adapted are elaborated. These techniques assist in realising and ameliorating on the results. Only the position at which correlation is maximum, is received. In order to obtain the run time differences, the position from maximum correlation calculation is multiplied with the period at which the sound was being sent. The temperature is a factor which has aided in realising a better result as the speed of sound in air; the used medium is dependent on the temperature of air at the moment of transmission. In **chapter five**, it is explained in detail and the relationship demonstrated.

The conclusion of the project is done in **chapter seven**. The project: software and Hardware for Position determination and display for an indoor Navigation System has been a success as the task has been tackled adequately. Some possible application of the project is outlined as well in chapter seven.
Last but not the least is **chapter eight** which explains most of the specific technical terms used in the entire project.

1.3- General setup

Figure 1.3 is a block diagram showing the general setup involving a foreign MSP430F169 supplying the run time differences though UART. These values are processed in the main MSP430F169 to obtain the coordinates of the object which is further represented on the matrix display board as a point. The control of this board is through SPI as seen in Figure 1.3.

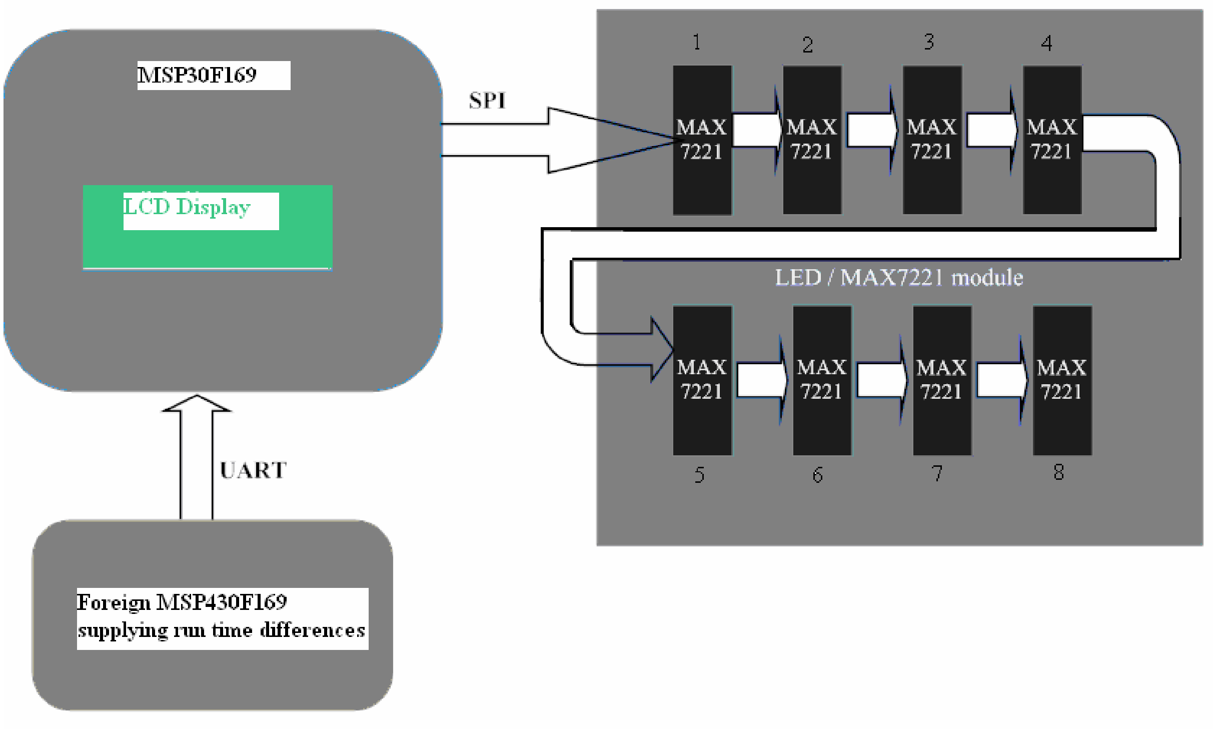


Figure 1.3: Block diagram of the general setup

Chapter 2: Theoretical Background

2.1 - From General Positioning System (GPS) to Local Positioning System (LPS

The GPS was created and realised by the American Department of Defense (DOD) and was originally based on and run with 24 satellites (21 satellites being required and 3 satellites as replacement). Nowadays, about 30 active satellites orbit the earth in a distance of 20200 km. GPS satellites transmit signals which enable the exact location of a GPS receiver, if it is positioned on the surface of the earth, in the earth atmosphere or in a low orbit. GPS is being used in aviation, nautical navigation and for the orientation ashore. Further it is used in land surveying and other applications where the determination of the exact position is required. The GPS signal can be used without a fee by any person in possession of a GPS receiver. The only prerequisite is an unobstructed view of the satellites (or rather of the sky). The correct name of the system is NAVSTAR (Navigation System for Timing and Ranging), but commonly, it is referred to as GPS (Global Positioning System). [16]
In GPS, there are 4 observables. These are:

I. Pseudo distance from the code measurement. Only this method is used in this project
II. Pseudo distance from integrated Doppler-Count
III. Distance from the phase carrier or carrier phase difference
IV. Difference in duration of signal from Interferometry measurements.

The code measurement is based on the correlation process; the PRN-Impulse series of a code finally over mounts the PRN-Impulse series produced by the receiver. The PRN is a binary signal with random noise-like properties which is generated by mathematical algorithm or "code", and consists of repeated pattern of 1's and 0's.
The phase of the reproduced code is shifted in time until the maximum correlation occurs.
The pseudo range is obtained from the formula;

$$p_i = r_i + c * \triangle t_i \qquad (2.1.0)$$

where r_i is the distance between receiver and satellite. This distance in LPS will be the distance from object (sound emitter) to the various sensors.
c is the speed of light in GPS and will be the speed of sound in air in LPS
t_i is the time difference between that of the satellite and the receiver in GPS meanwhile in LPS, it is the run time difference obtained from correlation. The geometrical representation of the setup in GPS could be seen in Figure 2.0.

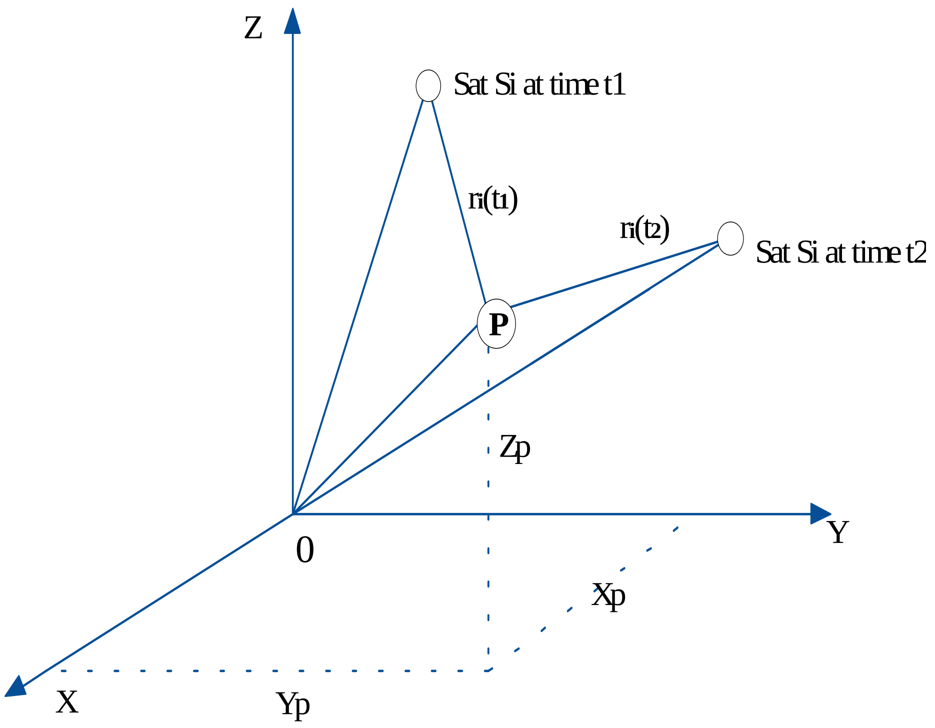


Figure 2.0: GPS setup with satellites and an object

P is the object whose position is to be determined.

The S_i are the locations of the satellites, $r_i(t_i)$ are the ranges between the corresponding satellite and the receiver (object), x_p, y_p and z_p are the Cartesian coordinates of the receiver.

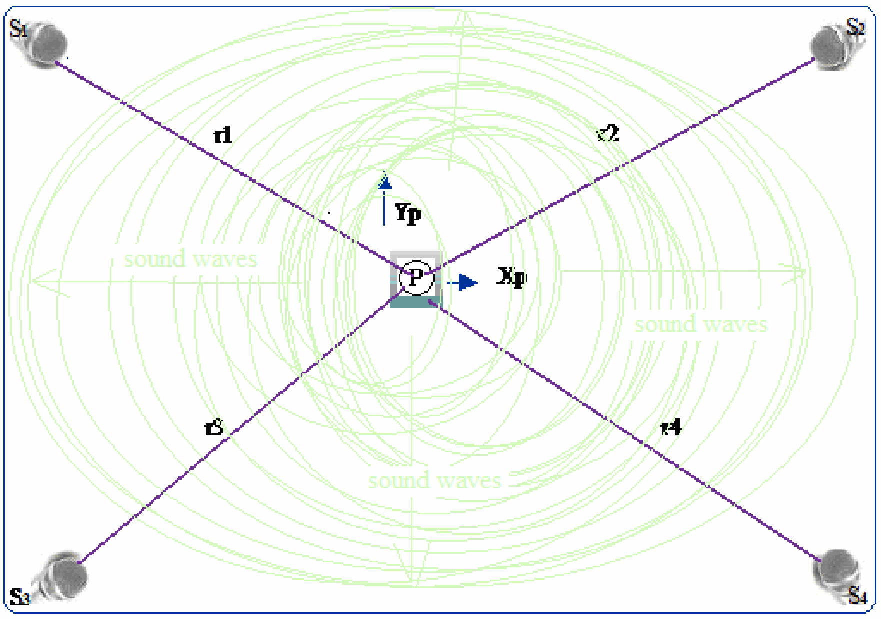


Figure 2.1: Mobile object and four sensors in LPS

The geometrical distance r_i from Satellite S_i in the GPS is represented by r_1 to r4. This distance is calculated using the formula of geometry below in equation 2.1.1.

$$r_i = \sqrt{[(x_i - x_p)^2 + (y_i - y_p)^2 + (z_i - z_p)^2]} \quad = \left| x_i - x_p \right| \qquad (2.1.1)$$

x_i, y_i , z_i are the x, y and z coordinates of the satellite in geometrical system in GPS meanwhile they are the Cartesian coordinates of the four receivers in the LPS.
x_p, y_p and z_p are the Cartesian coordinates of the object to be determined.
Since there is no synchronisation of the systems (sender and receivers) the speed of light and time difference play a vital role. For the LPS, it is the speed of sound that is considered since the signal is sound (ultrasound). From the above point of view, it implies that the actual distance will be modified as below:

$$p_i = r_i + c * \Delta t_i = \left| x_i - x_p \right| + c * \Delta t_i \qquad (2.1.2.0)$$
$$r_i = c * (t_i - t_0) \qquad (2.1.2.1)$$

where t_o is the time of emission of sound Δt_i is the time of reception of sound.
The term $c*t_i$ is also a distance due to the speed of light in air in GPS and the speed of sound in air in LPS.
c is the speed of light for GPS meanwhile it is the speed of sound in air. This value is 331.4 metres per second at 0°C. The speed of sound in dry air is given approximately by $c \approx 331.4 + 0.6\vartheta$ m/s.
Here, ϑ is the temperature of dry air and c the speed of sound. Meanwhile Δt_i is the time difference between the time of the satellite and that of the receiver in GPS. In LPS, Δt_i is the run time difference obtained through correlation carried out in a parallel project.
The ranges are obtained as follows:

$$r_1 = [(x_1 - x_p)^2 + (y_1 - y_p)^2 + (z_1 - z_p)^2]^{(1/2)} \qquad (2.1.3)$$
$$r_2 = [(x_2 - x_p)^2 + (y_2 - y_p)^2 + (z_2 - z_p)^2]^{(1/2)} \qquad (2.1.4)$$
$$r_3 = [(x_3 - x_p)^2 + (y_3 - y_p)^2 + (z_3 - z_p)^2]^{(1/2)} \qquad (2.1.5)$$

With the three equations above, the Cartesian coordinates of the object could be obtained if there was a common time (synchronisation of the sender and receivers) since there is no possibility of synchronization, a fourth equation is unavoidable in this case. This will be explained later in this chapter as it is necessary to first choose a method for determining the position. Three methods could be considered here.

The first method is **the closed solution form**. An equation is said to be a closed-form solution if it solves a given problem in terms of functions and mathematical operations from a given generally accepted set. For example, an infinite sum would generally not be considered closed-form. However, the choice of what to call closed-form and what not is rather arbitrary since a new "closed-form" function could simply be defined in terms of the infinite sum.
In this project, this method will be considered.

The second method is **the Kalman-Filtering method**. The fast Kalman filter (FKF), devised by Antti Lange (1941), is an extension of the Helmert-Wolf blocking (HWB) method from geodesy to real-time applications of Kalman filtering (KF) such as satellite imaging of the Earth. Kalman filters are an important software technique for building fault-tolerance into a wide range of systems, including real-time imaging. The computational advantage of **FKF** is marginal for applications using only small amounts of data in real-time data. Therefore improved built-in calibration and data communication infrastructures need to be developed first and introduced to public use before personal gadgets and machine-to-machine (M2M) devices can make the best out of **FKF**. [10]

The third method is **the iterative method (Taylor linearization).** This method will be employed as it is technically and easily realisable in microelectronics signal processing. It is also widely used in the industry.
The prerequisite is the assumption that $x_p{'}$, $y_p{'}$ and $z_p{'}$ are points very close to the awaited points, whereby the awaited points sum up to make the position of the object represented by $\mathbf{x_p}$, $\mathbf{y_p}$ and $\mathbf{z_p}$.
The range will further be calculated using the estimated position of the object. This is realised as shown in the equations below:

$$r_1{'} = [(x_1 - x_p{'})^2 + (y_1 - y_p{'})^2 + (z_1 - z_p{'})^2]^{(1/2)} \quad (2.1.6)$$

$$r_2{'} = [(x_2 - x_p{'})^2 + (y_2 - y_p{'})^2 + (z_2 - z_p{'})^2]^{(1/2)} \quad (2.1.7)$$

$$r_3{'} = [(x_3 - x_p{'})^2 + (y_3 - y_p{'})^2 + (z_3 - z_p{'})^2]^{(1/2)} \quad (2.1.8)$$

where $r_1{'}$ is the calculated range from estimated coordinates of the first sensor, $r_2{'}$ calculated range from estimated coordinates of the second sensor, $r_3{'}$ calculated range from estimated coordinates of the third sensor.
This approximation differs from the normal distance **r** with a delta $\triangle$- sum.
This further implies

$$\triangle r_1 = r_1 - r_1{'} \quad (2.1.9)$$

$$\triangle r_2 = r_2 - r_2{'} \quad (2.\ 2.0)$$

$$\triangle r_3 = r_3 - r_3{'} \quad (2.2.1)$$

From the above equations,
Δr_1 is the difference between the range of the first and the estimated range
Δr_2 is the difference between the range of the second and the estimated range
Δr_3 is the difference between the range of the third and the estimated range. The question here helps in obtaining the coordinates of the object. Since the position of the object had earlier been estimated, all that is necessary to be done is to add the difference to the estimated position for various axes, and it is now possible to move to the next stage.

$$\triangle x_p = x_p - x_p^{'} s \quad (2.2.2)$$

$$\triangle y_p = y_p - y_p^{'} \quad (2.2.3)$$

Δx is the difference in the x axis between the points on the real x axis and the estimated point on the axis whereas Δy is the difference in the y-axis between the point on the real y-axis and the estimated point on the axis.

The influence of the Δ is on all of the ranges. It is further assumed as follows:

$$\triangle r_1 = a_1 * \triangle x_p + b_1 * \triangle y_p + c_1 * \triangle z_p \quad (2.2.4)$$

$$\triangle r_2 = a_2 * \triangle x_p + b_2 * \triangle y_p + c_2 * \triangle z_p \quad (2.2.5)$$

$$\triangle r_3 = a_3 * \triangle x_p + b_3 * \triangle y_p + c_3 * \triangle z_p \quad (2.2.6)$$

Where a, b and c are the factors influencing the system and a is the factor influencing x directly, b is the factor influencing y directly, c is the factor influencing z directly.
These factors could be obtained as demonstrated below from equations 2.2.7 to 2.2.9.

$$a_i = -\frac{x_i - x_p'}{r_i'} \qquad (2.2.7)$$

$$b_i = -\frac{y_i - y_p'}{r_i'} \qquad (2.2.8)$$

$$c_i = -\frac{z_i - z_p'}{r_i'} \qquad (2.2.9)$$

The index *i* stand for 1 to 3 corresponding to the three equations involving the ranges **r**.
A matrix is then formed to determine the **Δ** of the coordinates.

$$\begin{bmatrix} \triangle r_1 \\ \triangle r_2 \\ \triangle r_3 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} * \begin{bmatrix} \triangle x_p \\ \triangle y_p \\ \triangle z_p \end{bmatrix} \qquad (2.3.0)$$

If the matrix A = $\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix}$ then

$$\triangle x = A^{-1} * \triangle r \qquad (2.3.1)$$

A^{-1} is the system matrix

It then implies $\begin{bmatrix} \triangle x_p \\ \triangle y_p \\ \triangle z_p \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix}^{-1} * \begin{bmatrix} \triangle r_1 \\ \triangle r_2 \\ \triangle r_3 \end{bmatrix}$ (2.3.2)

As seen earlier, in equations 2.2.2 and 2.2.3, the equations below representing the coordinates of the objects could be obtained.

$$x_p = x_p' + \triangle x_p \qquad (2.3.3)$$

$$y_p = y_p' + \triangle y_p \qquad (2.3.4)$$

$$z_p = z'_p + \triangle z_p \qquad (2.3.5)$$

It is assumed and precautions taken to see into it that the coordinates of the receivers are accurate. Up to now, only three equations have been used to determine Cartesian coordinates. Since there is a fourth unknown: time constant k a fourth equation will be necessary to obtain the x_p, y_p, z_p and k_i.

$$k_i = c * \triangle t_i \qquad (2.3.6)$$

With t_i being the time elapsed between sending and receiving of signal obtained through correlation and the period of the sent signal.
The system now has four equations which are as follows:

$$p_1 = [(x_1 - x_p')^2 + (y_1 - y_p')^2 + (z_1 - z_p')^2]^{(1/2)} + c * \triangle t_1 \qquad (2.3.7)$$

$$p_2 = [(x_2 - x_p')^2 + (y_2 - y_p')^2 + (z_2 - z_p')^2]^{(1/2)} + c * \triangle t_2 \qquad (2.3.8)$$

$$p_3 = [(x_3 - x_p^{'})^2 + (y_3 - y_p^{'})^2 + (z_3 - z_p^{'})^2]^{(1/2)} + c*\triangle t_3 \quad (2.3.9)$$

$$p_4 = [(x_4 - x_p^{'})^2 + (y_4 - y_p^{'})^2 + (z_4 - z_p^{'})^2]^{(1/2)} + c*\triangle t_4 \quad (2.4.0)$$

Transforming the four equations above in matrix form makes the system to be better interpreted in mathematical form.

$$\begin{bmatrix} \triangle p_1 \\ \triangle p_2 \\ \triangle p_3 \\ \triangle p_4 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 & 1 \\ a_2 & b_2 & c_2 & 1 \\ a_3 & b_3 & c_3 & 1 \\ a_4 & b_4 & c_4 & 1 \end{bmatrix} * \begin{bmatrix} \triangle x_p \\ \triangle y_p \\ \triangle z_p \\ \triangle k_p \end{bmatrix} \quad (2.4.1)$$

$$\mathrm{A_{new}} = \begin{bmatrix} a_1 & b_1 & c_1 & 1 \\ a_2 & b_2 & c_2 & 1 \\ a_3 & b_3 & c_3 & 1 \\ a_4 & b_4 & c_4 & 1 \end{bmatrix} \quad (2.4.1.0)$$

$$\triangle \mathrm{x} = \begin{bmatrix} \triangle x_p \\ \triangle y_p \\ \triangle z_p \\ \triangle k_p \end{bmatrix} \quad (2.4.1.1)$$

$$\triangle p = \begin{bmatrix} \triangle p_1 \\ \triangle p_2 \\ \triangle p_3 \\ \triangle p_4 \end{bmatrix} \quad (2.4.1.2)$$

That means

$$\triangle \mathrm{x} = A_{new}^{-1} * \triangle p \quad (2.4.2)$$

$$k = k^{'} + \triangle k_p \quad (2.4.3)$$

Where k is $c*\triangle t_i$

So the final equation to determine the Cartesian position of the object which will then be displayed by lighting up a corresponding LED on the LED matrix board is given by:

$$\underline{\underline{p_i = \sqrt{(x_i - x_p)^2 + (y_i - y_p)^2 + (z_i - z_p)^2} + c*\triangle t_i}} \quad (2.4.4)$$

$$p_i = c*(t_i - t_0) \quad (2.4.5)$$

t_0 is the time at which the signal is sent.

The unknowns are xp, yp and zp which are the Cartesian coordinate of the object in question x_i, y_i and z_i are estimated and c at the moment known, with the help of the voltage fall over a diode in MSP430F169 and $\triangle t_i$ is obtained from a parallel process.

The distance covered by sound in air is linearly related to the time taken at a particular temperature. Taking as an example the temperature of 25°C, a curve of distance covered by sound and time taken could be obtained as shown in Figure 2.2 below.

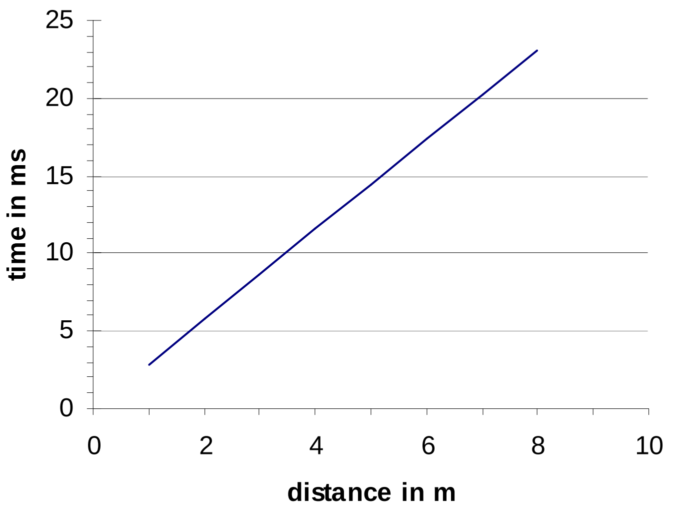


Figure 2.2: Distance and time covered by sound in air at 25°C

The next step is working out step by step the inverse a 4x4 matrix which requires that the determinant be first calculated. The next section has more details on inverse matrix determination.

2.2 - A 4x4 inverse matrix calculation

This section handles the task of obtaining the inverse of a 4x4 matrix step by step to ease the understanding and programming. The first step will be calculating the cofactors of the 4x4 matrix as described in section 2.2.1.

$$A^{-1} = (\frac{1}{\det(A)})(Co(A))^T$$

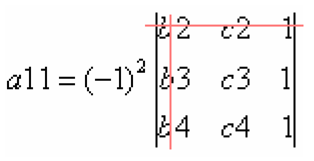
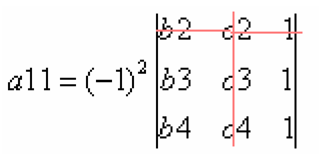
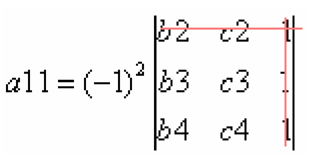
2.2 .1 - Calculating the cofactors of a 4x4 matrix

Let a matrix A as below be considered. The last column is filled with 1 to make up a 4x4 matrix due to the addition of a fourth constant being a fourth row. The system matrix must be square.

$$A = \begin{bmatrix} a1 & b1 & c1 & 1 \\ a2 & b2 & c2 & 1 \\ a3 & b3 & c3 & 1 \\ a4 & b4 & c4 & 1 \end{bmatrix} \qquad (2.4.5)$$

2. Calculating the first element a11 of the cofactors.

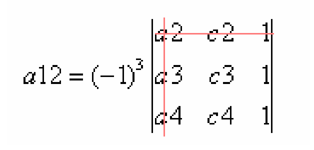
$$a11 = (-1)^2 \begin{vmatrix} b2 & c2 & 1 \\ b3 & c3 & 1 \\ b4 & c4 & 1 \end{vmatrix} \qquad (2.4.5.1)$$

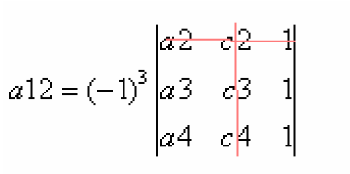




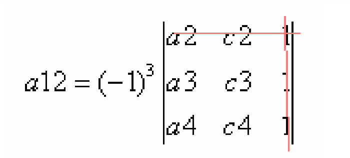
$$a11 = \left[b2(c3 - c4) - c2(b3 - b4) + (b3 * c4 - c3 * b4) \right] \qquad (\ 2.4.5.2)$$

3. Calculating the second element a12 of the cofactors.

$$a12 = (-1)^3 \begin{vmatrix} a2 & c2 & 1 \\ a3 & c3 & 1 \\ a4 & c4 & 1 \end{vmatrix} \qquad (2.4.5.3)$$





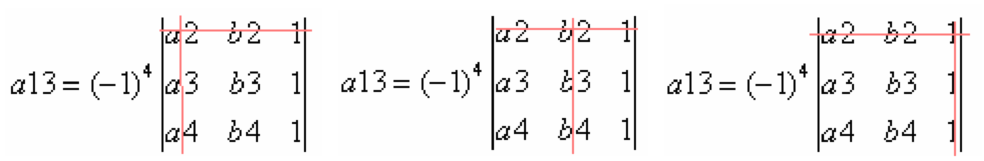


$$a12 = -\left[a2(c3 - c4) - c2(a3 - a4) + (a3 * c4 - a4 * c3) \right]$$

$$a12 \quad = -\left[a2(c3 - c4) - c2(a3 - a4) \right]$$

4. Calculating the third element a13 of the cofactors.

$$a13 = (-1)^4 \begin{vmatrix} a2 & b2 & 1 \\ a3 & b3 & 1 \\ a4 & b4 & 1 \end{vmatrix}$$



$$a13 \quad = \left[a2(b3 - b4) - b2(a3 - a4) + (a3 * b4 - a4 * b3) \right] \qquad (2.4.5.4)$$

The same procedure continues for determining the remaining 13 elements for the cofactors. Since the pattern is clear, only the results of the rest are listed below.

a14 = [a2(b3*c4 - b4*c3) - b2(a3*c4 - a4*c3) + c2(a3*b4 - b3*a4)]
a21 = -[b1(c3 - c4) - c1(b3 - b4) + (b3*c4 - b4*c3)]
a22 = [a1(c3 - c4) - c1(a3 - a4)+ (a3*c4 - a4*c4)]
a23 = -[a1(b3 - c4) - b1(a3 - a4) + (a3*b4 - a4*b3)]
a31 = [b1(c2 - c4) - c1(b2 - b4) + (b2*c4 – c2*b4)]
a32 = -[a1(c2 - c4) - c1(a2 - a4) + (a2*c4- a4*c2)]
a33 = [a1(b2 - b4) - b1(a2 - a4) + (a2*b4- a4*b2)]
a34 = -[a1(b2*c4 - b4*c2) - b1(a2*c4 - a4*c2) +c1(a2*b4- a4*b2)]
a41 = -[b1(c2 – c3) - c1(b2 – b3) + (b2*c3 – b3*c2)]
a42 = [a1(c2 - c3) - c1(a2 – a3) + (a2*c3- a3*c2)]
a43 = -[a1(b2 – b3) - b1(a2 – a3) + (a2*b3- a3*b2)]
a44 = [a1(b2*c3 – b3*c2) - b1(a2*c3 – a3*c2) +c1(a2*b3- a3*b2)]

Up to this step, all the values are known except the determinant of the 4x matrix `A(det(A)).` This will be carried out in the next step.

$$\det(A) = a1\begin{vmatrix} b2 & c2 & 1 \\ b3 & c3 & 1 \\ b4 & c4 & 1 \end{vmatrix} - b1\begin{vmatrix} a2 & c2 & 1 \\ a3 & c3 & 1 \\ a4 & c4 & 1 \end{vmatrix} + c1\begin{vmatrix} a2 & b2 & 1 \\ a3 & b3 & 1 \\ a4 & b4 & 1 \end{vmatrix} - \begin{vmatrix} a2 & b2 & c2 \\ a3 & b3 & c3 \\ a4 & b4 & c4 \end{vmatrix} \qquad (2.4.5.5)$$

a_i, b_i, c_i and 1 are the elements of the 4x4 matrix A

$$\begin{aligned} &= a1(b2\begin{vmatrix} c3 & 1 \\ c4 & 1 \end{vmatrix} - c2\begin{vmatrix} b3 & 1 \\ b4 & 1 \end{vmatrix} + \begin{vmatrix} a3 & 1 \\ a4 & 1 \end{vmatrix}) \\ &-b1(a2\begin{vmatrix} c3 & 1 \\ c4 & 1 \end{vmatrix} - c2\begin{vmatrix} a3 & 1 \\ a4 & 1 \end{vmatrix} + \begin{vmatrix} a3 & c3 \\ a4 & c4 \end{vmatrix}) \\ &+c1(a2\begin{vmatrix} b3 & 1 \\ b4 & 1 \end{vmatrix} - b2\begin{vmatrix} a3 & 1 \\ a4 & 1 \end{vmatrix} + \begin{vmatrix} a3 & b3 \\ a4 & b4 \end{vmatrix}) \end{aligned}$$

$$-(a2\begin{vmatrix} a3 & b3 \\ a4 & b4 \end{vmatrix} - b2\begin{vmatrix} a3 & c3 \\ a4 & c4 \end{vmatrix} + c2\begin{vmatrix} a3 & b3 \\ a4 & b4 \end{vmatrix})$$

The determinant of a square matrix is a single number calculated by combining all the elements of the matrix. This is obtained by multiplying the main diagonal and subtracting the product of the other diagonal. The difference is then multiplied with the outer elements. The results are then added with each other or subtracted from each other depending on the signs. The elements of the inverse matrix are then obtained as shown below:

$$Co(A) = \begin{bmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a31 & a32 & a33 & a34 \\ a41 & a42 & a43 & a44 \end{bmatrix} \qquad (2.4.6)$$

$$(Co(A))^T = \begin{bmatrix} a11 & a21 & a31 & a41 \\ a12 & a22 & a32 & a42 \\ a13 & a23 & a33 & a43 \\ a14 & a24 & a34 & a44 \end{bmatrix} \qquad (2.4.7)$$

$$A^{-1} = (\frac{1}{\det(A)}) \begin{bmatrix} a11 & a21 & a31 & a41 \\ a12 & a22 & a32 & a42 \\ a13 & a23 & a33 & a43 \\ a14 & a24 & a34 & a44 \end{bmatrix} \qquad (2.4.8)$$

Equation 2.4.8 could now be incorporated into equation 2.4.1 to obtain the equation below.

$$\begin{bmatrix} \triangle x_p \\ \triangle y_p \\ \triangle z_p \\ \triangle k_p \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 & 1 \\ a_2 & b_2 & c_2 & 1 \\ a_3 & b_3 & c_3 & 1 \\ a_4 & b_4 & c_4 & 1 \end{bmatrix}^{-1} * \begin{bmatrix} \triangle p_1 \\ \triangle p_2 \\ \triangle p_3 \\ \triangle p_4 \end{bmatrix} \qquad (2.4.5.4)$$

where $\begin{bmatrix} a_1 & b_1 & c_1 & 1 \\ a_2 & b_2 & c_2 & 1 \\ a_3 & b_3 & c_3 & 1 \\ a_4 & b_4 & c_4 & 1 \end{bmatrix}^{-1}$ is A^{-1}

2.3.1 - The closed solution method in obtaining xp, yp and t0

After analyzing the *iterative* method (Taylor linearization), the closed solution method is worth analysing as the theory is more comprehensive and the programming behind less complex.
The coordinate equation of an object as seen earlier is considered

$(x_i - x_p)^2 + (y_i - y_p)^2 = C^2(t_i - t_0)^2$

The index i stands for the four sensors (that is from 1 to 4). All other variables and constants remain as in the method before.
Expanding the above equation gives;

$$x^2_1 - 2x_1x_p + x_p{}^2 + y^2_1 - 2y_1y_p + y_p{}^2 = C^2t_1{}^2 - 2C^2t_1t_0 + C^2t_0{}^2 \qquad (2.5.1)$$

$$x^2_2 - 2x_2x_p + x_p{}^2 + y^2_2 - 2y_2y_p + y_p{}^2 = C^2t_2{}^2 - 2C^2t_2t_0 + C^2t_0{}^2 \qquad (2.5.2)$$

$$x^2_3 - 2x_3x_p + x_p{}^2 + y^2_3 - 2y_3y_p + y_p{}^2 = C^2t_3{}^2 - 2C^2t_3t_0 + C^2t_0{}^2 \qquad (2.5.3)$$

$$x^2_4 - 2x_4x_p + x_p{}^2 + y^2_4 - 2y_4y_p + y_p{}^2 = C^2t_4{}^2 - 2C^2t_4t_0 + C^2t_0{}^2 \qquad (2.5.4)$$

Equation 2.5.4 will now be subtracted from 2.5.1, 2.5.2 and 2.5.3.

$$(x^2_1 - x^2_4) + (y^2_1 - y^2_4) - 2(x_1 - x_4)x_p - 2(y_1 - y_4)y_p = C^2(t_1{}^2 - t_4{}^2) - 2C^2(t_1 - t_4)t_0 \qquad (2.5.5)$$

$$(x^2_2 - x^2_4) + (y^2_2 - y^2_4) - 2(x_2 - x_4)x_p - 2(y_2 - y_4)y_p = C^2(t_2{}^2 - t_4{}^2) - 2C^2(t_2 - t_4)t_0 \qquad (2.5.6)$$

$$(x^2_3 - x^2_4) + (y^2_3 - y^2_4) - 2(x_3 - x_4)x_p - 2(y_3 - y_4)y_p = C^2(t_3{}^2 - t_4{}^2) - 2C^2(t_3 - t_4)t_0 \qquad (2.5.7)$$

Looking for the value of t_0 in equation 2.5.5.

$$2C^2(t_1{}^2 - t_4{}^2)t_0 = 2(x_1 - x_4)x_p + 2(y_1 - y_4)y_p - [(x^2_1 - x^2_4) + C^2(t_1{}^2 - t_4{}^2)]$$

$$\Rightarrow \quad t_0 = \frac{-(x^2_1 - x^2_4) + 2(x_1 - x_4)x_p - (y^2_1 - y^2_4) + 2(y_1 - y_4)y_p + C^2(t_1{}^2 - t_4{}^2)}{2C^2(t_1 - t_4)} \qquad (2.5.8)$$

From equation 2.5.8, the following constants are regrouped.

A:= $(t_1 - t_4)$

B:= $2(x_1 - x_4)$

C:= $2(y_1 - y_4)$

D:= $[(x^2_1 - x^2_4) + (y^2_1 - y^2_4) - C^2(t_1{}^2 - t_4{}^2)]$

$$\Rightarrow \quad t_0 = \frac{Bx_p + Cy_p - D}{A} \qquad (2.5.9)$$

A second t_0 is obtainable from equation 2.5.6.

$$2C^2(t_2^{\ 2}-t_4^{\ 2})t_0 = 2(x_2-x_4)x_p + 2(y_2-y_4)y_p - [(x^2_{\ 2}-x^2_{\ 4}) + C^2(t_2^{\ 2}-t_4^{\ 2})]$$

$$\Rightarrow \quad t_0 = \frac{-(x^2_{\ 2}-x^2_{\ 4}) + 2(x_2-x_4)x_p - (y^2_{\ 2}-y^2_{\ 4}) + 2(y_2-y_4)y_p + C^2(t_1^{\ 2}-t_4^{\ 2})}{2C^2(t_2-t_4)} \qquad (2.6.1)$$

From equation 2.6.1, the following constants are regrouped. It is to be noted that the constant $2C^2$ disappears since i is a constant factor that will always appear on both sides of the equations for t_0.

A1:= (t_2-t_4)

B1:= $2(x_2-x_4)$

C1:= $2(y_2-y_4)$

D1:= $[(x^2_{\ 2}-x^2_{\ 4}) + (y^2_{\ 2}-y^2_{\ 4}) - C^2(t_2^{\ 2}-t_4^{\ 2})]$

$$\Rightarrow \quad t_0 = \frac{B_1x_p + C_1y_p - D_1}{A_1} \qquad (2.6.2)$$

Equating equation 2.5.8 to 2.6.2

$$\Rightarrow \quad \frac{Bx_p + Cy_p - D}{A} = \frac{B_1x_p + C_1y_p - D_1}{A_1}$$

$$\Rightarrow \quad A_1(Bx_p + Cy_p - D) = A(B_1x_p + C_1y_p - D_1)$$

$$x_p = \frac{y_p(AC_1 - A_1C) + A_1D - AD_1}{A_1B - AB_1} \qquad (2.6.3)$$

Using another pair of t_0, another x$_p$ could be obtained. Considering equation 2.5.7 above, the next t_0 could be obtained as below.

$$2C^2(t_3^{\ 2}-t_4^{\ 2})t_0 = 2(x_3-x_4)x_p + 2(y_3-y_4)y_p - [(x^2_{\ 3}-x^2_{\ 4}) + C^2(t_3^{\ 2}-t_4^{\ 2})]$$

$$\Rightarrow \quad t_0 = \frac{-(x^2_{\ 3}-x^2_{\ 4}) + 2(x_3-x_4)x_p - (y^2_{\ 3}-y^2_{\ 4}) + 2(y_3-y_4)y_p + C^2(t_3^{\ 2}-t_4^{\ 2})}{2C^2(t_3-t_4)}$$

(2.6.4)

From equation 2.6.1, the following constants are regrouped.

A2:= (t_3-t_4)

B2:= $2(x_3-x_4)$

C2:= $2(y_3-y_4)$

D2:= $[(x^2_{\ 3}-x^2_{\ 4}) + (y^2_{\ 3}-y^2_{\ 4}) - C^2(t_3^{\ 2}-t_4^{\ 2})]$

$$\Rightarrow \quad t_0 = \frac{B_2x_p + C_2y_p - D_2}{A_2} \qquad (2.6.5)$$

A fourth t_0 is obtained so as to equate the next two t_0 to obtain another equation dependent on x_p and y_p as unknown.

$$2C^2(t_1^2 - t_2^2)t_0 = 2(x_1 - x_2)x_p + 2(y_1 - y_2)y_p - [(x^2_1 - x^2_2) + C^2(t_1^2 - t_2^2)]$$

$$\Rightarrow \quad t_0 = \frac{-(x^2_1 - x^2_2) + 2(x_1 - x_2)x_p - (y^2_1 - y^2_2) + 2(y_1 - y_2)y_p + C^2(t_1^2 - t_2^2)}{2C^2(t_1 - t_2)} \qquad (2.6.6)$$

From equation 2.6.1, the following constants are regrouped.

A3:= $(t_1 - t_2)$
B3:= $2(x_1 - x_2)$
C3:= $2(y_1 - y_2)$
D3:= $[(x^2_1 - x^2_2) + (y^2_1 - y^2_2) - C^2(t_1^2 - t_2^2)]$

$$\Rightarrow \quad t_0 = \frac{B_3 x_p + C_3 y_p - D_3}{A_3} \qquad (2.6.7)$$

Now, equating equation 2.6.5 to 2.6.7, it is possible to obtain x_p dependent on y_p.

$$\frac{B_2 x_p + C_2 y_p - D_2}{A_2} = \frac{B_3 x_p + C_3 y_p - D_3}{A_3}$$

$$\Rightarrow \quad A_3(B_2 x_p + C_2 y_p - D_2) = A_2(B_3 x_p + C_3 y_p - D_3)$$

$$x_p = \frac{y_p(A_2 C_3 - A_3 C_2) + A_3 D_2 - A_2 D_3}{A_3 B_2 - A_2 B_3} \qquad (2.6.8)$$

Equations 2.6.3 and 2.6.8 are equal.

$$\frac{y_p(AC_1 - A_1 C) + A_1 D - AD_1}{A_1 B - AB_1} = \frac{y_p(A_2 C_3 - A_3 C_2) + A_3 D_2 - A_2 D_3}{A_3 B_2 - A_2 B_3} \qquad (2.6.9)$$

From equation 2.6.8, the following constants are regrouped.

E:= $AC_1 - A_1 C$
F:= $A_1 D - AD$
G:= $A_2 C_3 - A_3 C_2$
H:= $A_3 D_2 - A_2 D_3$
I:= $A_1 B - AB_1$
J:= $A_3 B_2 - A_2 B_3$

Now equation 2.6.9 could be rewritten as

$$\frac{y_pE+F}{I} = \frac{y_pG+H}{J} \tag{2.7.0}$$

$$\Rightarrow \quad J(y_pE+F) = I(y_pG+H)$$

$$\boxed{y_p = \frac{IH-JF}{JE-IG}} \tag{2.7.1}$$

y_p could now be substituted in one of the equations above involving x_p, for example equation 2.6.8. Once these two unknowns (y_p and x_p) have been obtained, the third unknown (t_0) could be obtained by substituting the values of y_p and x_p in one of the four equations involving x_p, y_p and t_0, for example equation 2.6.7. It is also possible to obtain the three unknowns (x_p, y_p and t_0) using just three equations instead of four. With the three unknowns determined, it is now left for the software to adequately represent these values (y_p and x_p) on the 16x32 matrix LED board.

2.3.2 - An alternative closed form solution.

In this method, it is considered that the four positions of the microphones are such that they form a rectangle.

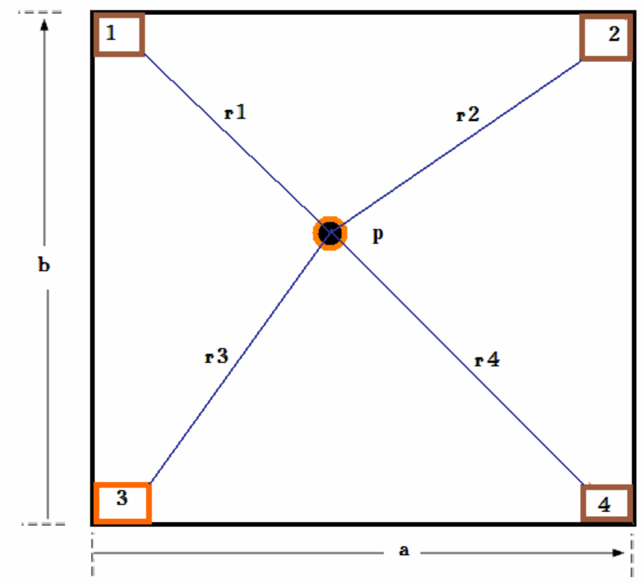


Figure 2.3.1: Rectangular surface of experiment

Let the surface be rectangular. That means the distance between the point 1 to 2 and 3 to 4 is the same, with the value a and the distance between the point 1 to 3 and 2 to 4 is the same, with the value b.

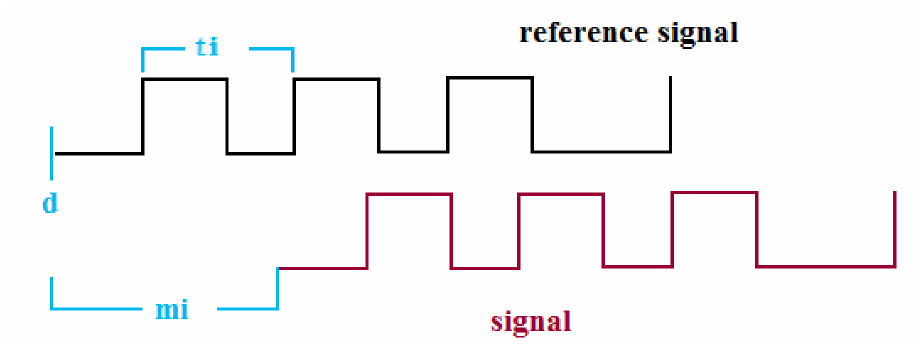


Figure 2.3.2: Signal and reference signal showing run time difference through correlation.

On Figure 2.3.2, `ti` stands for the time obtained from maximum correlation for the `i`-position and `d` is an offset time since the sender and receiver work asynchronically.

$$m_i = t_i + d \qquad (2.7.2)$$

m_i is a time corresponding to the sum of the offset time and the run time.

$$r_i = t_i * C \qquad (2.7.3)$$

r_i are the ranges as represented in Figure 2.3.1 above and obtained as in equation 2.7.3 with C being the speed of sound, dependent on the temperature.

$$x_2 = x_4$$
$$x_2 = x_1 + a$$
$$x_2 = x_3 + a \quad \Rightarrow \quad x_3 = x_2 - a \quad = x_4 - a$$
$$y_2 = y_1$$
$$y_2 = y_4 + b \quad \Rightarrow \quad y_4 = y_2 - b$$
$$y_2 = y_3 + b$$
$$y_3 = y_2 - b$$

The following equations have been mentioned already in the first method and it is the general equation for determining the position of an object in a room.

$$x^2{}_1 + y^2{}_1 - r^2{}_1 = 0 \qquad (2.7.4)$$
$$x^2{}_2 + y^2{}_2 - r^2{}_2 = 0 \qquad (2.7.5)$$
$$x^2{}_3 + y^2{}_3 - r^2{}_3 = 0 \qquad (2.7.6)$$
$$x^2{}_4 + y^2{}_4 - r^2{}_4 = 0 \qquad (2.7.7)$$

Subtracting (2.7.6) from (2.7.7)

$$\Rightarrow \quad x^2{}_4 - x^2{}_3 + y^2{}_4 - y^2{}_3 - r^2{}_4 + r^2{}_3 = 0$$
$$x^2{}_2 - (x_2 - a)^2{}_3 + (y_2 - b)^2 - (y_2 - b)^2 - r^2{}_4 + r^2{}_3 = 0$$
$$2x_2 a - a^2 - C^2 t^2{}_4 + C^2 t^2{}_3 = 0$$

It is known that t = (m-d)

$$\Rightarrow \quad \boxed{x_2 = \frac{a^2 + C^2(m_4 - d)^2 - C^2(m_3 - d)^2}{2a}} \qquad (2.7.8)$$

The value of x_2 obtained in equation 2.7.8 above is the first x-coordinate of the object with the assumptions taken earlier. The value of the corresponding y_2 value could be obtained by subtracting (2.7.5) from (2.7.4) as follows.

$$\Rightarrow \qquad x^2{}_4 - x^2{}_2 + y^2{}_4 - y^2{}_2 - r^2{}_4 + r^2{}_2 = 0$$

$$x^2{}_4 - x^2{}_4 + y^2{}_2 - (y_2 - b)^2 + r^2{}_4 - r^2{}_2 = 0$$

$$2y_2 b - b^2 + C^2 t^2{}_4 - C^2 t^2{}_2 = 0$$

It is known that t = (m-d)

$$\Rightarrow \qquad \boxed{y_2 = \frac{b^2 + C^2(m_2 - d)^2 - C^2(m_4 - d)^2}{2b}} \qquad (2.7.9)$$

The pair of values (x_2 and y_2) obtained above is just a solution out of many. This could be other solutions should give the same coordinate. Due to the quadratic nature of the equation, a value may lie out of the bounded rectangle. In this case, it is not part of the solution.

	(2.7.4)	**(2.7.5)**	**(2.7.6)**	**(2.7.7)**
- (2.7.4)	0	x_1	y_1	x_1, y_2
- (2.7.5)	x_1	0	x_1, y_1	y_1
- (2.7.6)	y_1	x_1, y_1	0	x_1
- (2.7.7)	x_1, y_1	y_1	x_1	0

Table 2.3.1: second closed form method solutions

As shown in Table 2.3.1, the rest of the solutions could be obtained by subtracting the various equations from one another. Some of them may not return real solutions. Such cases are examined closely and rejected as a solution.

2.4 - Comparison of Display technologies or systems

After obtaining the coordinates of the object, the next point is choosing an adequate display system. There is a good variety of display technology, some of which are CRT, PDP (Plasma Display Panel), LCD, SED, LCoS including LED Display. After a brief analysis of the existing displays, it was found out that some were unnecessarily expensive, some difficult to find, some not suited for a battery powered system, some unnecessarily complicated to electrically control and some unsuitable in terms of size for an indoor navigation system. LED Display system was finally chosen as it has many characteristics favourable for controlling and displaying a 2 dimensional coordinates of an object. The characteristics that make the LED suitable for such a display are outlined in the general characteristics section of LED. The most important reason for choosing LED as a display was due to the visibility of LED lights at longer distances, making it adequate as a display system. Most typical LEDs are designed to operate with not more than 30–60 Miliwatt of electrical power. One of the major reasons for choosing LED Display is its high efficiency, as measured by its light output per unit power input. Another point is the fact that the solid package of the LED can be designed to focus its light while on the other hand incandescent and fluorescent sources often require an external reflector to collect light and direct it in an applicable manner. Further more, LEDs, being solid state components, are difficult to be damaged with external shock. LEDs light up very quickly. A typical red LED will achieve full brightness in microseconds. An example is the Philips Lumileds DS23 [11] with less than 100ns. LEDs are very small in size and many could fit onto printed circuit boards. There are of course a few disadvantages of using LEDs of which are minimal compared to the advantages. One of the disadvantages is that their performance largely depends on the ambient temperature of the operating environment. Over-driving the LED in high ambient temperatures may result in overheating of the LED package, consequently proceeding to device failure. Also, LEDs have to be supplied with the correct current which could involve resistors in series or current-regulated power supplies. The LED Display used requires only one resistor per segment. This resistor could also be used to vary the brightness of the LED. The brightness of the LED could be controlled digitally using the brightness register which varies from 0 to 15, with 0 being minimum brightness and 15, maximum brightness.

Chapter 3: MSP430 F169 Starter Kit and used Peripheries

3.1 - General Characteristics

After analyzing the method to determine the coordinates and a display technology chosen, an appropriate microcontroller will be necessary. The MSP430 has been chosen for many reasons, which will be outlined in this chapter.
The MSP430 family of ultra-low-power 16-bit RISC mixed-signal processors from Texas Instruments (TI) provides the ultimate solution for battery-powered measurement applications. It is designed for low cost, low power consumption embedded applications. It suits well for this project as the system (stand alone) uses battery meanwhile the mixed-signal and digital technologies are fully utilised. This mixed-signal ability permits system designers to simultaneously interface to analogue signals, sensors and digital components while maintaining the low power consumption. They can run up to 8MHz (up to 16 MHz for the new MSP430F2xxx series) and their consumption is only 250 µA per MIPS (Million instructions per second), which makes them perfect for portable and handheld devices. The DMA controller module transfers data from one address to another without CPU intervention.

Typical features of the MSP430F169STK include:

- 16-bit RISC architecture
- 125 ns Instruction Cycle
- 48 I/O pins
- 60 Kilobytes Flash
- 2 Kilobytes RAM
- 3-channel Internal DMA
- 12-bit A/D Converter with Internal Reference, Sample-and-Hold, and auto scan
- Dual 12-bit D/A Converters with Synchronization
- 16-bit Timer A with 3 Capture/Compare Registers
- 16-bit Timer with 7 Capture/Compare-With-Shadow Registers
- On-chip Comparator
- USART0 Functions as Asynchronous UART, Synchronous SPI, or I2C

The USART0 is used here for the synchronous communication of the MSP430 board with the MAX7221 and LED modules. It is the Serial Periphery Interface (SPI) mode. The clock is required to latch out the data from MSP430 to the chip of the MAX7221. The chip select (CS) has to be low for a particular chip to permit data to be written to it.

- USART1 functions as Asynchronous UART or Synchronous SPI. In this project, USART1 is used in the asynchronous communication with another MSP430 to obtain the point of maximum correlation which reflects the time elapsed between sending and receiving signals.
- Brown-out Detection
- Supply Voltage Supervisor/Monitor with Programmable Level Detection
- Programmable Code Protection by Security Fuse
- On board LCD used in displaying input, temperature and general information.
- Three buttons B1, B2 and B3 for input purpose

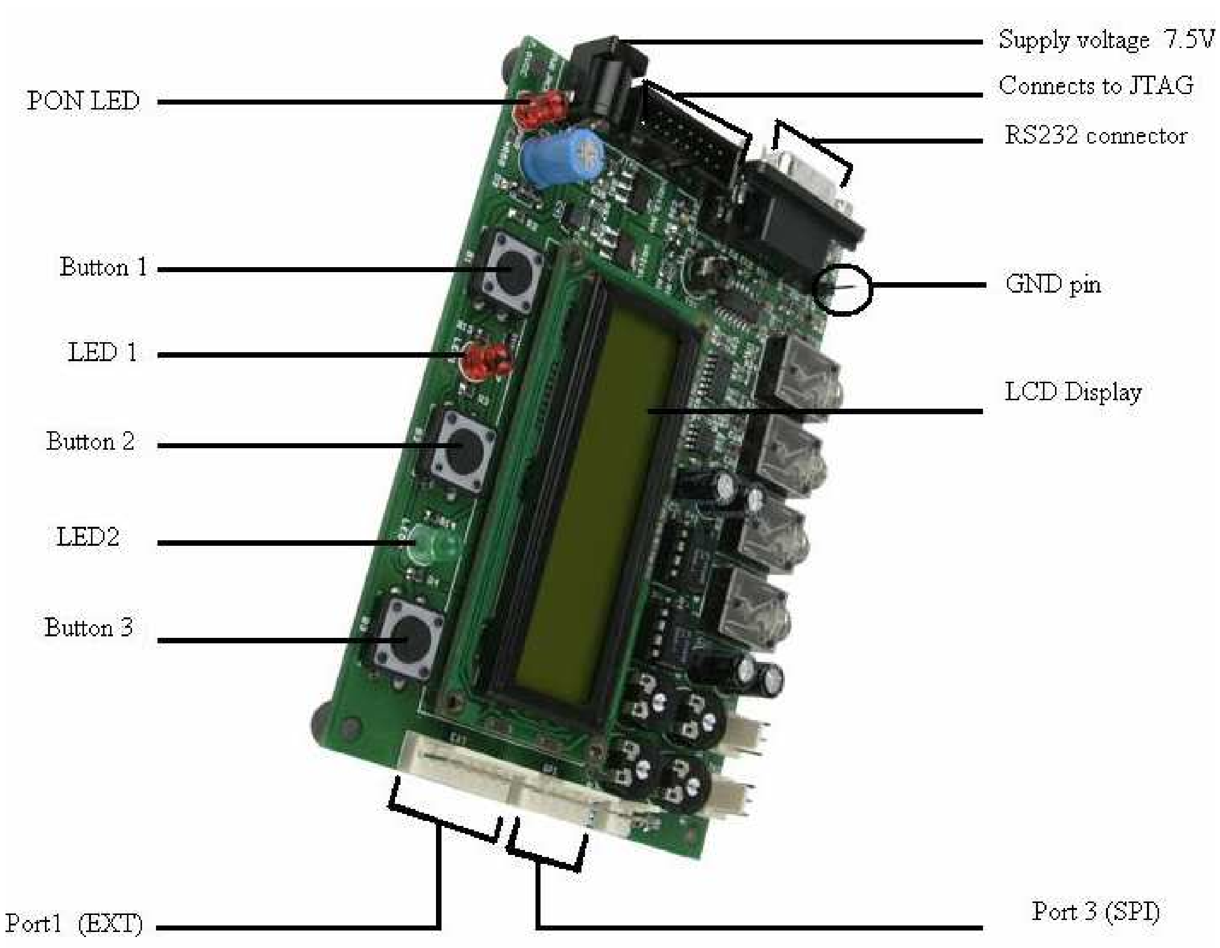


Figure 3.1.1: MSP430F169 starter Kit and used peripheries

Figure 3.1.1 shows MSP430F169 Starter Kit with its peripheries. It has an LCD display which can display up to 32 characters on two lines. There are three buttons B1, B2 and B3 on board assisting in inputs. The c code is loaded into the MSP430 from a computer through a JTAG connector. It is supplied with 7.5 V DC. It has 3 LEDs: PONLED is the power-on LED indicating the connection to power, LED1 is a red LED between B1 and B2 and there is LED2, which is green and found between B2 and B3.

MSP430 has an extra-low power architecture (Figure 3.1.2: MSP430 Architecture overview), 1.8 – 3.6V operation, 6µs wakeup from standby mode, 16 bits ALU and many other characteristics making it suitable for a project like this one, needing a battery powered system. A 16-bit Reduced instruction set computer (RISC) CPU, peripherals and flexible clock system are combined by using the von-Neumann common memory address bus and memory data bus. Figure 3.1.2 below gives more explanation to this structure. The MSP430 offers solutions for present and future mixed signal applications.

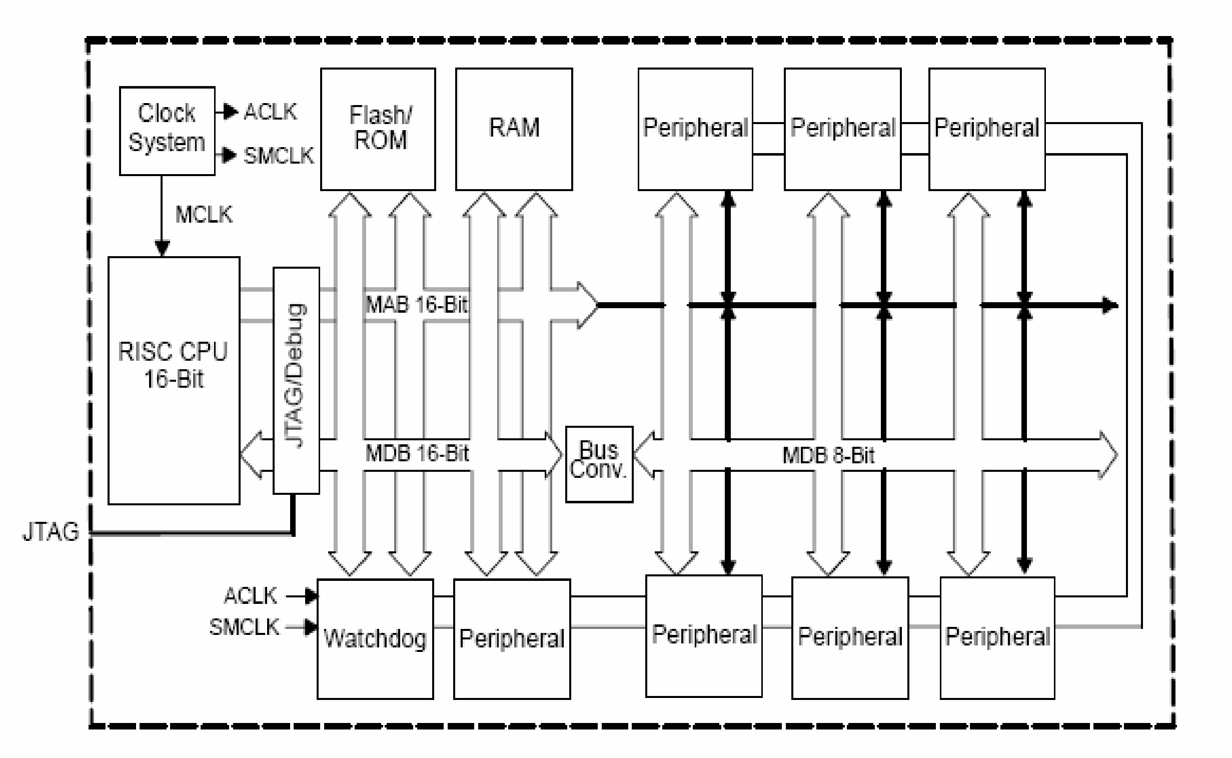


Figure 3.1.2: MSP430 Architecture overview [9]

It has a uniform instruction format, using a single word with the OPCODE in the same bit positions in every instruction, demanding less decoding and simple addressing modes, less complex addressing performed through sequences of arithmetic and /or load-store operations. Less decoding reduces the time taken to interpret code and consequently implying a faster reaction or processing.

3.2 – Ports of the MSP430F169

There are all together six ports on the MSP430F169 Starter Kit. The most used port in this project is the port 3. It is used in lighting up the controllable LEDs available on the board. It is equally used in the SPI communication as well as in the UART communication to receive run time differences from a parallel project.
As is standard on microcontrollers, most pins connect to a more specialized peripheral, but if that peripheral is not needed, the pin may be used for general-purpose I/O. The pins are divided into 8-bit groups called "ports", each of which is controlled by a number of 8-bit registers.

P*x*IN
Port *x* input. This is a read-only register, and reflects the current state of the pin.

P*x*OUT
Port *x* output. The values written to this read/write register are driven out the corresponding pins when they are configured to output.

P*x*DIR
Port *x* data direction. Bits written as 1 configure the corresponding pin for output. Bits written as 0 configure the pin for input.

P*x*SEL
Port *x* function select. Bits written as 1 configure the corresponding pin for use by the specialized peripheral. Bits written as 0 configure the pin for general-purpose I/O. Port 0 ('3xx parts only) is not multiplexed with other peripherals and does not have a P0SEL register.

P*x*IES
Port *x* interrupt edge select (ports 0-2 only). Selects the edge, which will cause the P*x*IFG bit to be set. When the input bit changes from matching the P*x*IES state to not matching it (that is whenever a bit in P*x*IES XOR P*x*IN changes from clear to set), the corresponding P*x*IFG bit is set.

P*x*IFG
Port *x* interrupt flag (ports 0-2 only). Set whenever the corresponding pin makes the state change requested by P*x*IES. This can be cleared only by software (Can also be set by software).

P*x*IE
Port *x* interrupt enable (ports 0-2 only). When this bit and the corresponding P*x*IFG bit are both set, an interrupt is generated.

Note that some pins have special purposes either as inputs or outputs. (For example, timer pins can be configured as capture inputs or PWM outputs.) In this case, the P*x*DIR bit controls which of the two functions the pin performs even when the P*x*SEL bit is set. If there is only one special function, then P*x*DIR is generally ignored. [9]

3.3 - UART and SPI

There are two main communication protocols involved with the outside of the microcontroller. They are the Universal Asynchronous Receiver/ Transmitter (UART) and the Serial Peripheral Interface protocol. The MSP430F169 has the ability of offering these two protocols at the same time, as it has a USART0 and USART1, configurable to work at the same time. The UART is used in the time difference reception from a parallel process in another MSP430F169. This is in a continuous mode.

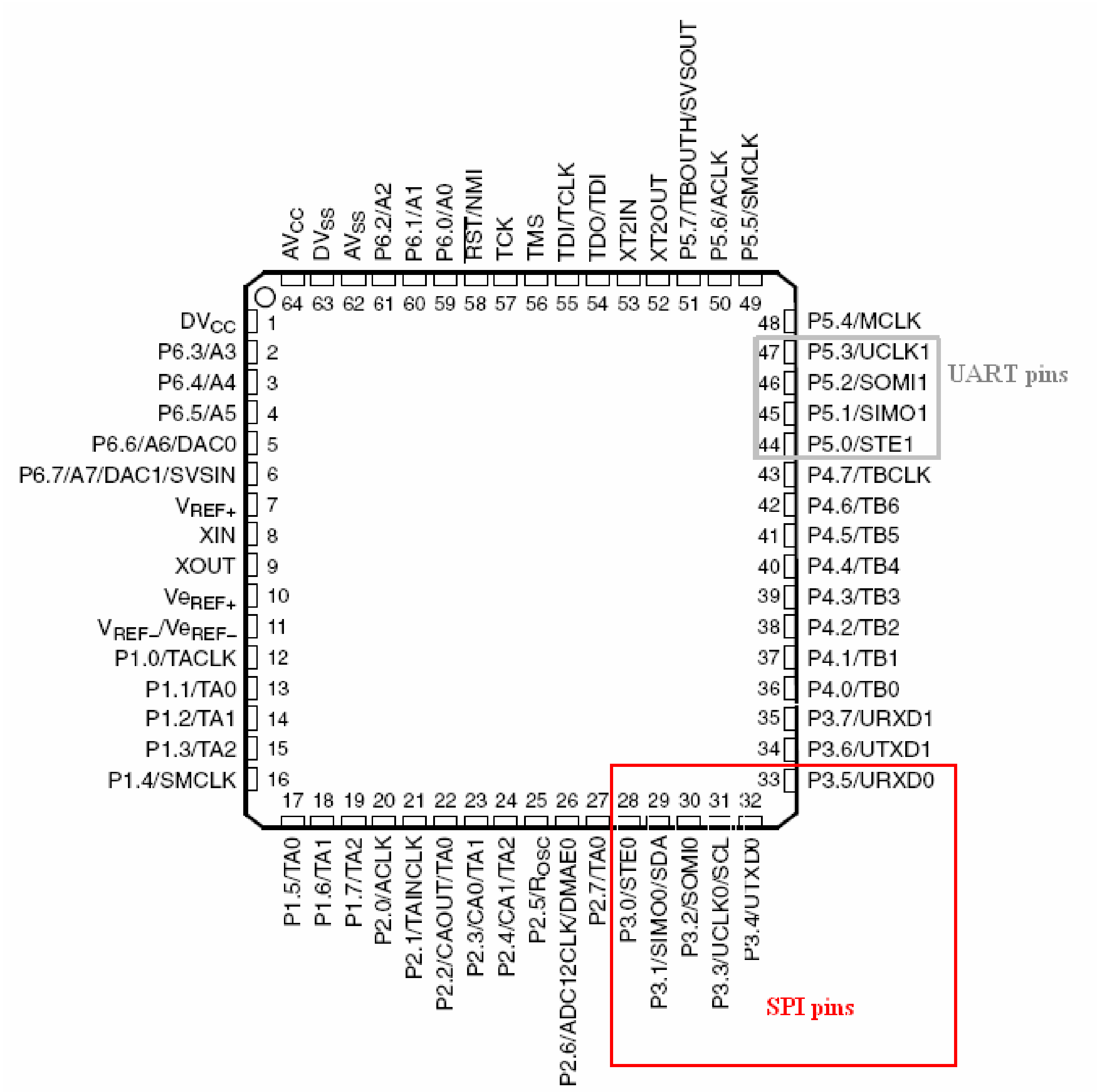


Figure 3.3.0: Relevant pins of MSP430 for UART use

The SPI is used in controlling the LEDs with the help of MAX7221 chips. The USART1 module is used in communicating with devices and systems that support the RS232 communications [28].

3.3.1 UART

USART as UART is used in this project in receiving data from another MSP403 running parallel with each other. This communication is duplex, continuous and rapid. The programming of this section is thoroughly explained and illustrated in Chapter five.

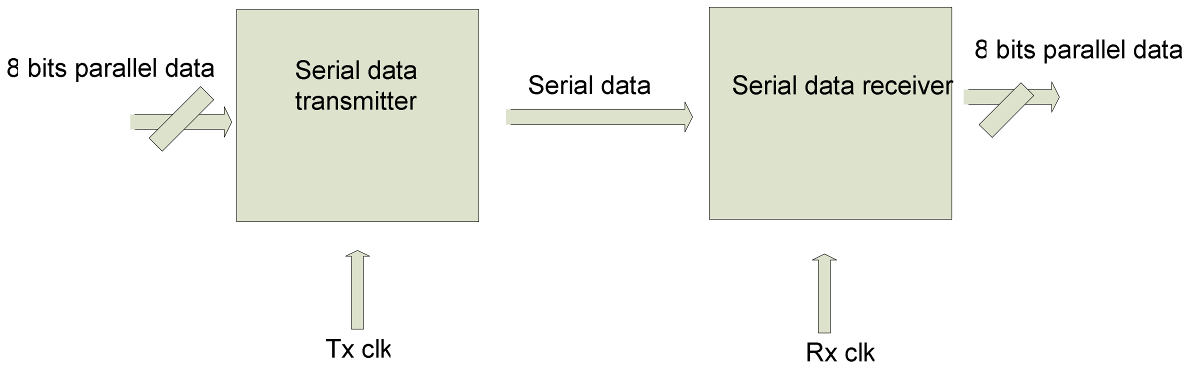


Figure 3.3.1: Block diagram of UART communication

3.3.2 - SP I

SPI stands for serial periphery interface. It is a synchronous communication needing a clock to permit the data move out of the microprocessor sending.
In synchronous mode, the USART connects the MSP430 to an external system via three or four pins: SIMO, SOMI, UCLK, and STE. SPI mode is selected when the SYNC bit is set and the I2C bit is cleared.

The features of SPI include:
- 7- or 8-bit data length.
  This setting was the most convenient as far the communication interface and speed was concerned.
- 3-pin and 4-pin SPI operation
- Master or slave modes
- Independent transmit and receive shift registers
- Separate transmit and receive buffer registers
- Selectable UCLK polarity and phase control.
  The clock is generated by the master. This is also true in full duplex communication where the master transmits and receives likewise the slave.
- Programmable UCLK frequency in master mode
  The UCLK frequency can be altered and this is possible only in the master mode as already explained.
- Independent interrupt capability for receive and transmit

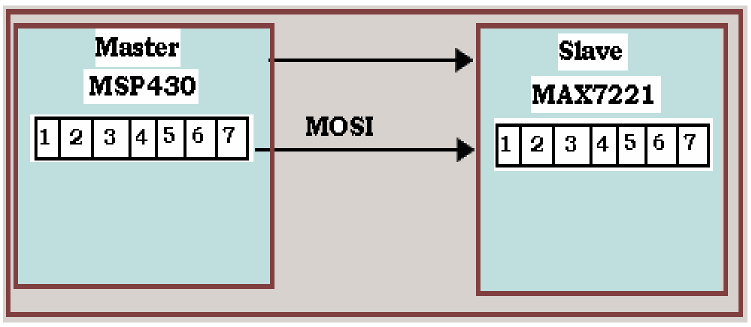


Figure 3.3.2: MOSI block diagram

MOSI (master output, slave input) is used in realising a functional and frictionless communication between the microcontroller and the 8 MAX7221 slaves.

The master is MSP430F169 sending out data synchronically to the slave (MAX7221). Data is latched out using the clock of the master. A detailed description is given in the software section in Chapter five.
It is very important to follow a specific order in the re-configuration process of this module. Failure to do that may result in anomalous behaviour without explanations. These steps and the orders are as follows:
1) Set SWRST (BIS.B #SWRST, &UxCTL)
The SWRST (software reset) configuration is done in the control register UxCTL with x being 0 or 1. The two modes could be operated separately and at the same time.
2) Initialize all USART registers with SWRST=1 (including UxCTL)
3) Enable USART module via the Mex and SFRs (USPIEx)
4) Clear SWRST via software (BIC.B #SWRST, & UxCTL). Clearing SWSRT allows operation.
5) Enable interrupts (optional) via the IEx SFRs (URXIEx and/or UTXIEx).
The USART has one interrupt vector for transmission and one interrupt vector for recaption. The UTXIFGx interrupt flag is set by the transmitter to indicate that UxTXBUF is ready to accept another character. An interrupt request is generated if UTXIEx and GIE are also set. [9]

3.3.2.1- SPI Timing

The polarity and phase of UCLK are independently configured via the CKPL and CKPH control bits of the USART. Figure 3.3.3 below gives more details to the timing functionality of the SPI.

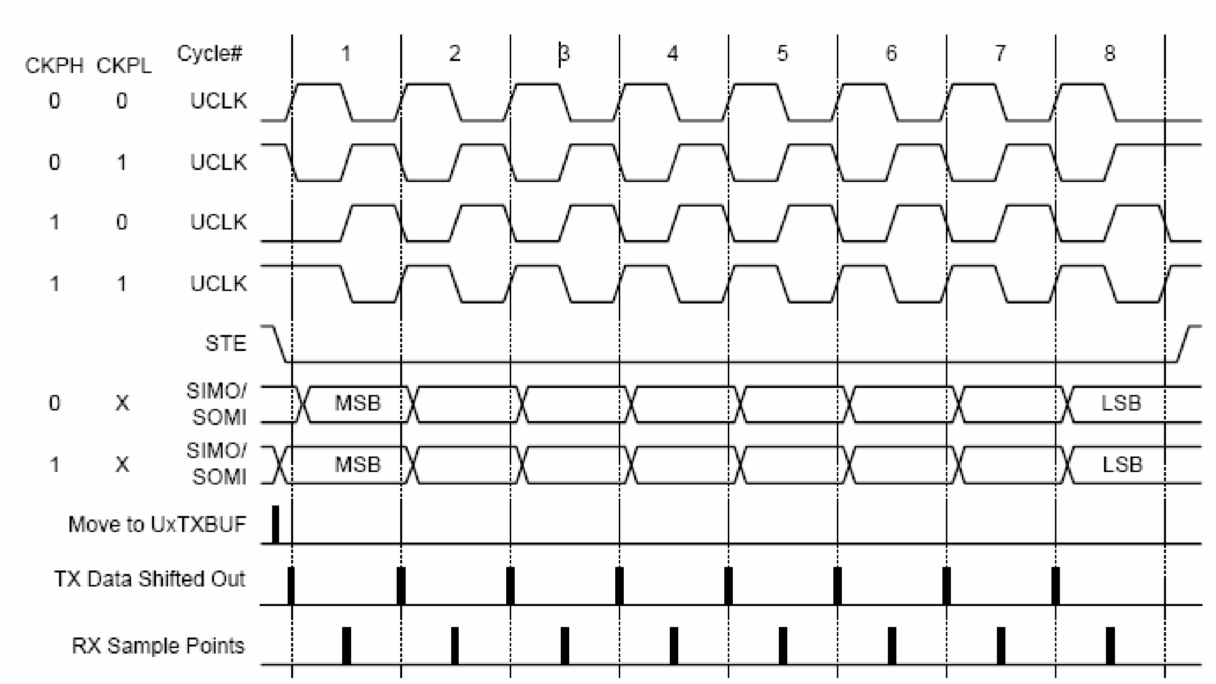


Figure 3.3.3: USART SPI Timing [9]

3.4 - Analogue-Digital Converter (ADC)

The ADC12 module supports fast, 12-bit analogue-to-digital conversions. The module implements a 12-bit SAR core, sample select control, reference generator and a 16 word conversion-and-control buffer. The conversion and control buffer allows up to 16 independent ADC samples to be converted and stored without any CPU intervention.

ADC12 features include:

- Greater than 200 ksps maximum conversion rate
- Monotonic 12-bit converter with no missing codes
- Sample-and-hold with programmable sampling periods controlled by software or timers.
- Conversion initiation by software, Timer_A, or Timer_B
- Software selectable on-chip reference voltage generation (1.5 V or 2.5 V)
- Software selectable internal or external reference
- Eight individually configurable external input channels
- Conversion channels for internal temperature sensor, AVCC, and external references
- Independent channel-selectable reference sources for both positive and negative references
- Selectable conversion clock source
- Single-channel, repeat-single-channel, sequence, and repeat-sequence conversion modes
- ADC core and reference voltage can be powered down separately
- Interrupt vector register for fast decoding of 18 ADC interrupts.
- 16 conversion-result storage registers

The ADC units on some MSP include a temperature sensor. This feature is easy to use and also accurate (typically within a degree). ADC12 is a single 12-bit analogue-to-digital converter, with a built-in sample-and-hold circuit. The front end consists of a multiplexer circuit which allows the developer to select one of eight external pins, or one of four internal sources.

The 8 external and four internal analogue signals are selected as the channel for conversion by the analogue input multiplexer. The input multiplexer is a break-before-make type to reduce input-to-input noise injection resulting from channel switching as shown in Figure 3.3.1

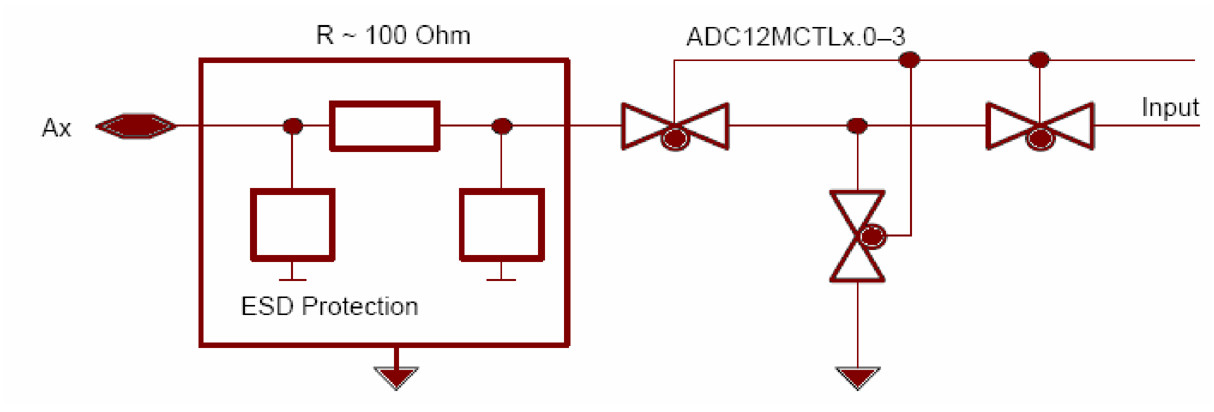


Figure 3.4.1: Multiplexer circuit [9]

There is an internal diode which permits the ADC to provide an estimated idea of operating temperature. The temperature diode varies with device. It is remarkably less accurate than an external temperature sensor though a low cost possibility when the microcontroller is already available. The MSP430F169 family has the volt-°C relation as follows: 3.55mV change in voltage is equivalent to 1°C. These values could be obtained as typical values from the data sheet. A more accurate value of temperature, it is better to use an external sensor. Since an accuracy of approximately 1°C is quit sufficient to obtain good results as far as the speed of sound is concerned, it is practical using this internal sensor. The above statement could be explained using the formula below:

$$C = 331+0.6*(\theta\pm1°C) \qquad (3.3.2)$$

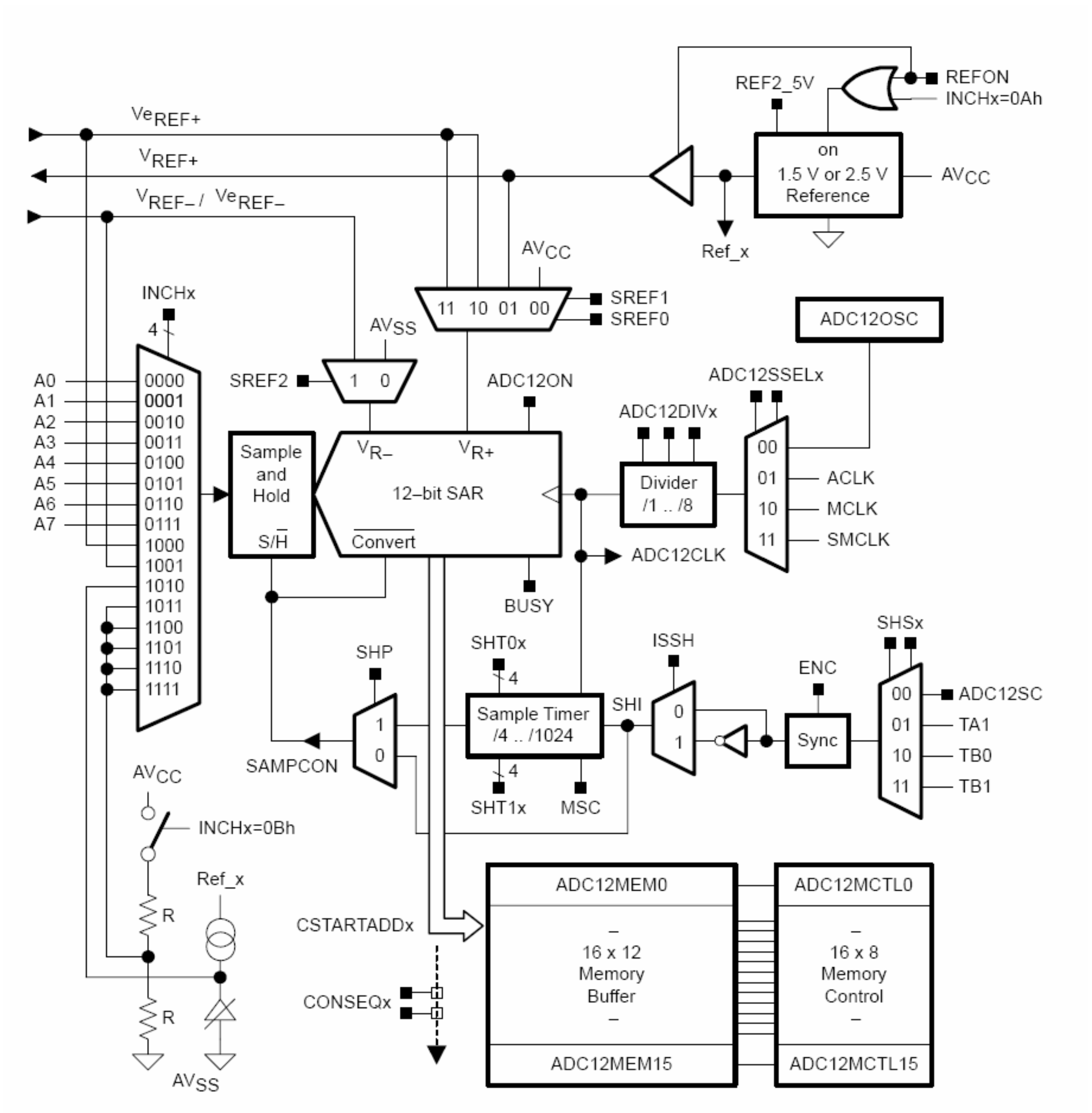


Figure 3.3.2: ADC12 block diagram [9]

The ADC core converts an analogue input to its 12-bit digital representation and stores the result in the conversion memory. The core uses two programmable/selectable voltage levels (V_{R+} and V_{R-}) to define the upper and lower limits of the conversion. The digital output

(NADC) is full scale (0xFFF≡ 4095 in decimal notation), when the input signal is equal to or higher than V_{R+}, and zero when the input signal is equal to or lower than V_{R-}. The input channel and the reference Voltage levels (V_{R+} and V_{R-}) are defined in the conversion-control memory.
The conversion formula for the ADC result NADC is:

$$N_{ADC} = 4095 * \frac{V_{in} - V_{R-}}{V_{R+} - V_{R-}} \tag{3.3.1}$$

The ADC12 core is configured by two control registers, ADC12CTL0 and ADC12CTL1. The core is enabled with the ADC12ON bit. The ADC12 can be turned off when not in use to save power. With few exceptions the ADC12 control bits can only be modified when ENC = 0. ENC must be set to 1 before any conversion can take place.

There are four conversion modes:

- **Single channel one-shot**

It is a single conversion with results being saved in one of the ADCMEM (analogue digital memory) registers.

- **Single channel repeated.**

It repetitively performs conversion until stopped. Results are being stored in the same ADCMEM register. Here, the typical method is to loop process when the BUSY flag clears. This mode will be explained in details in this chapter as it is the employed mode in this project.

- **Multiple channel, single sequence.**

In this mode, the ADC performs multiple conversions, looping through a specific number of ADCMEM registers one time.

- **Multiple channel repeated.**

It is identical to the previous case, apart from the fact that the series of conversion is repeated until stopped.
These mentioned modes have the advantage that they are a “start and forget" process. The code could be written such that the process is initialized and the code could perform other tasks while the conversion is underway. Nevertheless, these modes have the disadvantage or rather limitation in the fact that in the repeated mode, the software needs to be ready to read the ADCMEM registers before they are rewritten, or an interrupt will be generated.
Timing is performed by the conversion clock. This clock may be sourced by any of the clocks from the Basic Clock Module, or by a fixed RC oscillator which is a dedicated portion of the ADC. This oscillator is quit similar to DCO, with similar accuracy. The timer has to be initialized in the ADC12CTL1 register, and the DIV value and clock source must be selected so as to the conversion frequency meets the data sheet specifications. The conversion takes thirteen cycles of the conversion clock source.

ADC12 Control Registers

ADC12CTL0, ADC control register = 0. Address: 0x01A0h. All bits are read/write. Bits 15 though 4 may only be edited when ENC = 0.

SHT1/SHT0: Sample and Hold Time. SHT1 determines the sample and hold time for ADC0 through ADC7 and SHT0 determines the sample and hold time for ADC8 through ADC15. The sample and hold time is `4*(ADC_Clock_Time)*(n). [9]`
- AD12CTL1, ADC control register = 1. It has the address: 0x01A2. All bits except 0 (BUSY) are read or write. Bit 15 through 3 may only be edited when ENC = 0.

Bit	CSAdd3	CSAdd2	CSAdd1	CSAdd0	SHS1	SHS0	SHP	ISSH
Reset value	0	0	0	0	0	0	0	0
Bit Position	15(MSB)	14	13	12	11	10	9	8

Table 3.3.1: Control registers for ADC with reset value and bit position [9].

A single channel is sampled and converted continuously. The ADC results are written to the ADC12MEMx defined by the CSTARTADDx bits. Since only one ADC12MEMx memory is available for this mode, the results need to be read after each conversion and before the next sequence.
After conversion, the values of voltage change with reference of either 1.5V or 2.5V are saved in ADC12MEM10 register. The value is related to temperature as below:

$$U = T * 3.55\text{mV} + 986\text{mV} \qquad (3.3.2)$$
$$T_{1..5V} = \text{ADC12MEM10}*0.103158 - 277.75\ °\text{C} \qquad (3.3.3)$$
$$T_{2.5V} = \text{ADC12MEM10}*0.172 - 277.75\ °\text{C} \qquad (3.3.4)$$

Equation 3.3.2 is based on the reference voltage of 1.5V and equation 3.3.4 on the reference Voltage of 2.5V.
These relationships are explained in detail in the datasheet of MSP430-169. In order to obtain accurate values, a calibration at a temperature of about 25°C is necessary.

3.5- Liquid Crystal Display (LCD)

The temperature of the chip, obtained through ADC12, which is linearly dependent on the change of voltage in a diode of the circuit, is represented on the LCD. The setting of the coordinates of the axis of the four receivers is also displayed on the LCD. Writing a character to the liquid crystal of the MSP430F169 family is done in two steps. The first step is to send a command to set the LCD and the next step is to send the upper nibble and then the lower nibble of the data to be displayed. Figure 3.5.1 shows how the 32 characters are distributed.

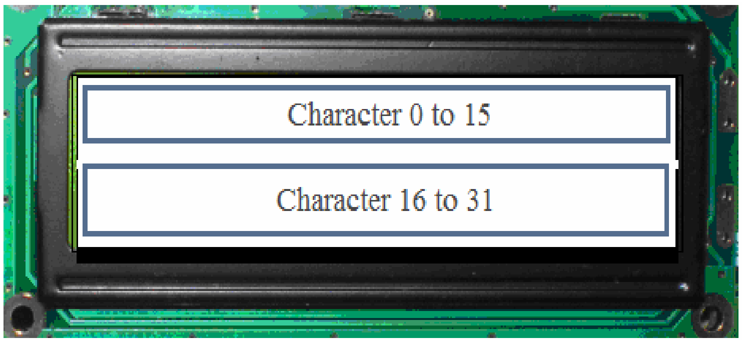


Figure 4.5.1: LCD of MSP430F169 showing the positioning of 32 characters.

As already mentioned, the LCD on MSP430F169STK can display 32 characters at the same time, with 16 characters per line as shown in Figure 3.2.5.1 above.
Only characters can be represented on the LDC. In order to represent integers for example, it is first converted into ASCII before sent to the LCD.

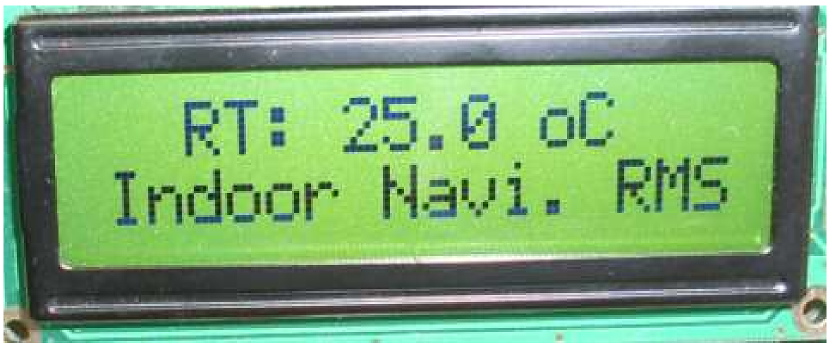


Figure 3.5.2: LCD of MSP430F169 showing displayed summer room temperature

Only half a byte is sent at a time that is the upper and lower nibble. Before sending the data, a command is first sent to set the LCD ready to accept the data. In sending the command, the upper nibble is first obtained by carrying out an AND operation of a constant with 0xF0. As such, the lower nibble is set to zero and the upper nibble to 1 or high. This is saved in temp. An OR operation is carried out between LDC data (P4OUT) and temp in order to position the content of temp, which is the upper nibble in LCD data without changing the content. The LCD is then set to command mode `(P4OUT& = ~BIT3)` which is represented as RS_LOW. At this point, the LCD is toggled `(_E();).` `_E();` is a subroutine that sets `P4OUT | = BIT1 (E_HIGH)` , then does nothing and then sets `P4OUT& = ~BIT1 (E_LOW).` After that, the lower nibble of the sent data (e) is obtained by shifting the content of **temp** 4 places to the right and carrying out an OR operation with LCD data. The e may be a data to clear the display, precise an address or any other reasonable operation. The toggle sub function `(_E();)` is then called again. This then completes the `SEND_CMD (e)` function. The details of the computing of the above, is explained in chapter five.

It is first of advisable to delay a bite before starting the sending operation. This whole sending process is identical to that of sending a command with the mare difference that the character sent now will be displayed on the LCD. For a better understanding, refer back to the subroutine `SEND_CMD();.` As explained earlier, only characters could be sent and recognized on the LCD. A trial to send integers directly, for example will result to unknown signs or characters that have no relation with what has been sent. After a few trials, it was noticed that the function `itoa` in mspgcc with Eclipse could not return the ASCII characters expected. For that reason, sprintf is used. The characters are saved in a buffer of type char and the content of the buffer is then sent to the LCD using the function named `SEND_CHAR ()`. An example is `sprintf(buffer,"%d",value)` converts to decimal base, `sprintf(str, "%x" ,value)` converts to hexadecimal base and `sprintf(str, "%o", va- lue)` converts to octal base. The data processed (for example 27°C) to send to LCD are merely integer and character. The temperature values are integer, degree (°) and Celsius (C) are both characters. The coordinates of the receivers are all integers. These values are put in through the buttons B1, B2 and B3 and are displayed as well on the LCD.

Chapter 4: Hardware Development

4.1 – Requirements

A few hardware and hardware component were necessary to realise this project. Some of them included:

- MSP430 F169 Starter Kit
- 8 chips of type MAX7221
- 8 pieces of 8x8 LED dot matrix display from Everlight Electronics Co., LTD model no: ELM-1883SRWA
- 16 Capacitors (8 of 10µF and 8 of 100nF)
- 8 resistors (each 25 KΩ or 22 KΩ)
- 7.5 V voltage supply for MSP430 F169
- 5 V voltage supply for MAX7221 connected at pin 19
- A few binders and cable of different sizes and types
- Printable circuit board, designed using Eagle

4.2 -Tool: Eagle

EAGLE (Easily Applicable Graphical Layout Editor) is an ECAD program. The Version 4.11 was used in this project. This software is relatively easy to learn and design simple electrical layouts. It provides a schematic editor, for designing circuit diagrams and an integrated PCB layout editor, which automatically starts off with all of the components required by the schematic. It also provides a good autorouter, which once the components have been placed will attempt to automatically find an optimal track layout to make the electrical connections. It does not always manage to find a way of routing all the signals, although it permits manual routing of critical paths such as power and high frequency lines before letting the autorouter handle the other connections.

4.3 – PCB Design for LED module

A printable circuit board (PCB) has been developed as an interface between the chips and the LEDs mainly.
It was necessary to abide as much as possible to certain rules to improve on the design and functionality of the circuit. Cable bends of 90° and less were avoided as much as possible ,although the circuit is in a relatively low frequency section. Cable bends of 90° or less could lead to data loss or corruption.
The capacitors were as closed to the chips (MAX7221) as possible, as such reducing unnecessary increase in conducting length and consequently increase in resistance.
An increase in the resistance leads to a greater voltage fall. This voltage fall is exactly what reduces the functionality of the capacitors, as their function is to store charges.
There are two possible ways of designing this layout in Eagle.
The first method is modular. Since there are 8 similar modules involved, only one circuit is designed and 8 of them reproduced. It was double sided; the front side for the LEDs and the other side for the other electrical components, like the resistor, capacitors as well as the connectors.

Five cables are connected from the MSP430F169 Starter Kit (STK) to the first module. The first cable is that for the clock to assist in latching out the data. This clock is serially connected to the rest of the modules and using connectors from one module to the next though the connections through the modules are internal on the printable circuit board (PCB).

The second cable is the data-in (DIN) cable conducting the data and addresses through the PCB to the MAX7221 integrated circuit and consequently setting some registers, lighting up or turning off specific LEDs as well as lighting a segment or a digit just to name a few.

The third cable is supply voltage (5V) cable supplying all the 8 modules. This cable is not from the microcontroller as it cannot meet up with the current requirement of the 8 LED displays.
The fourth cable is the Ground cable assisting in realizing a common earth for the modules, the microcontroller and the source. The Ground of the microcontroller is linked to that of the supply (5 volts) voltage source to realize the above statement.

The fifth cable is the chip select (CS) cable. The chip select is low active for MAX7221 permitting writing on to a chip while this method is easy to realize in Eagle since it does not involve a lot of drawing, therefore reducing the time factor. This method requires at the end relatively more soldering resulting from the connectors (joints) from one module to the next. In addition to that, when all the modules were placed next to each other, the LED displays could not fit to each other as there was a space of about 4 mm between the one display and the next, although the layout was in its minimal design size. See Figure4.3.2 on the next page. More so, a more complicated structure was required to keep the modules intact.
A block diagram of the first design is shown in figure 4.2.1 below. It is a unit out of eight. The attachment section should be consulted for the schematics.

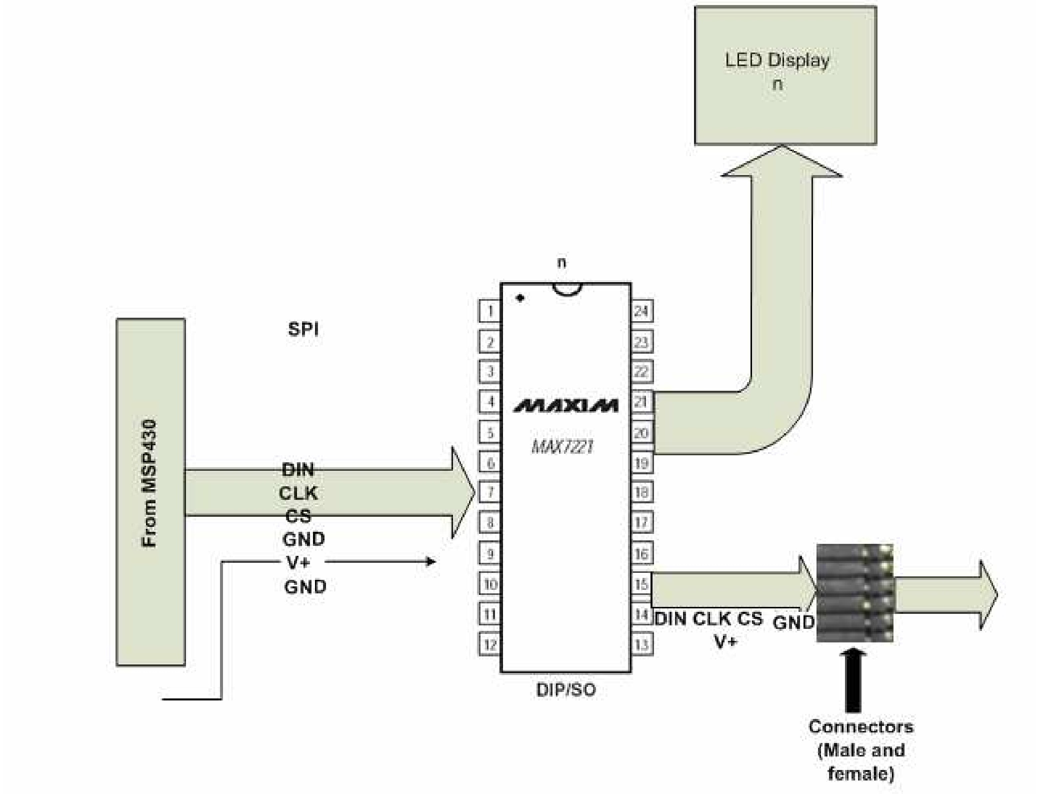


Figure 4.3.1: A block diagram of the first PCB design

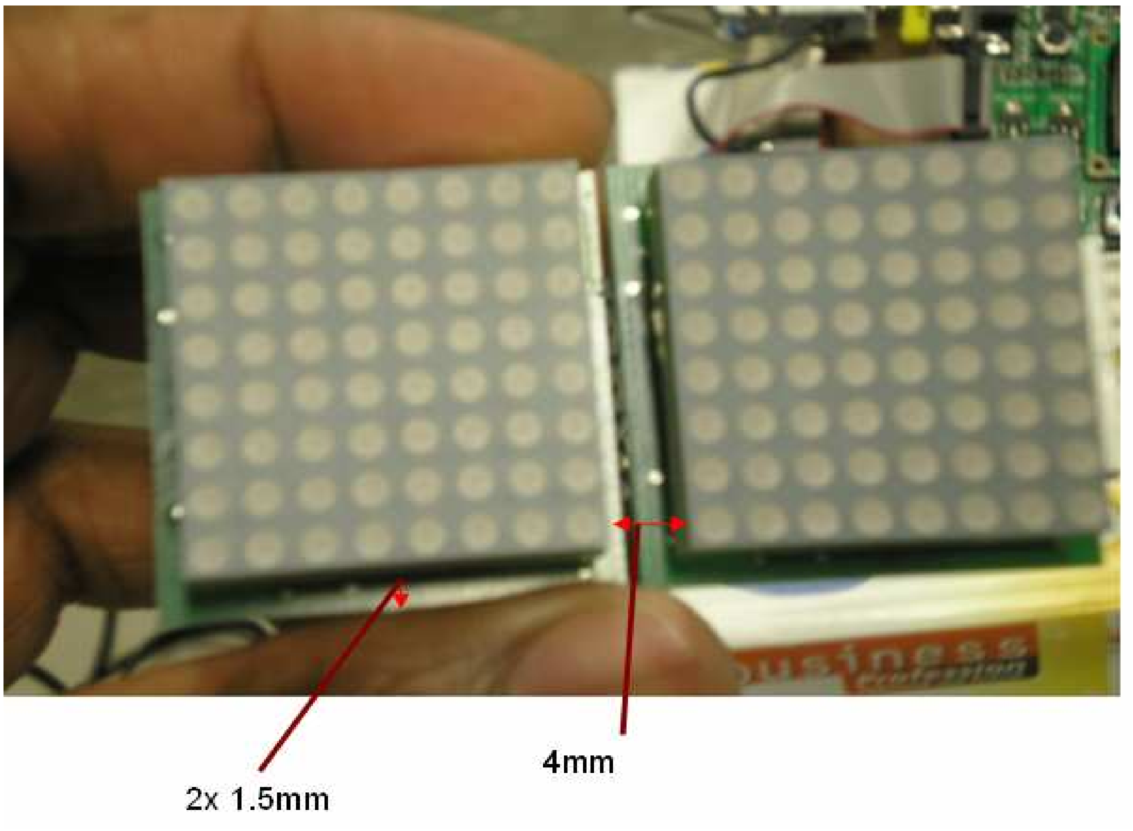


Figure 4.3.2: First design LED module and PCB mounted

Figure 4.3.2, shows two modules of the first design of PCB plugged together. The connector from one module to the next, being an additional hardware, is one of the disadvantages of the design since it involves 16 additional hardware pieces (connectors: 8 males and 8 females).

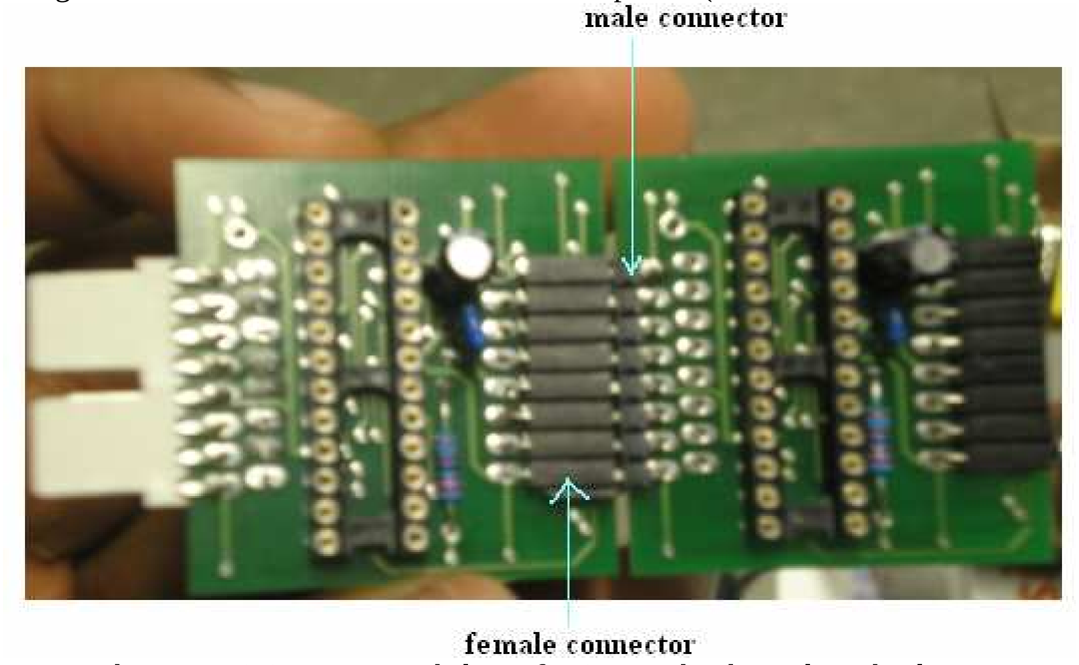


Figure: 4.3.3: Two modules of PCB and other electrical components

The table below shows how the pins of MAX7221 were connected to that of the LED and to the MSP430F169.

MSP430F169	MAX7221	SEG/ DIG/ Others	LED pin
P3.1	1	DIN	-
-	2	DIG0	13
-	3	DIG4	6
Port1 GND	4	GND	-
-	5	DIG6	15
-	6	DIG2	4
-	7	DIG3	10
-	8	DIG7	16
Port1 GND	9	GND	-
-	10	DIG5	11
-	11	DIG1	3
P3.0	12	$\overline{CS}$	-
P3.3	13	CLK	-
-	14	SEGA	14
-	15	SEGF	2
-	16	SEGB	8
-	17	SEGG	5
-	18	ISET	-
-	19	V+	-
-	20	SEGC	12
-	21	SEGE	7
-	22	SEGDP	9
-	23	SEGD	1
-	24	DOUT	-

Table 4.3.1: Pin to pin connection of MSP430, LED and MAX7221

There are four connections from the MSP430. The data is sent out (MOSI) through port 3 on pin 1. The ground from the controller is linked to that of MAX7221 IC at pin 4 and 9 through port 1 GND. Chip select is on port 3 pin 0 (connected to pin 12 of the IC) and the clock is on port 3 pin 3, connected to pin 13 of the IC.

Figure 4.3.4 on the next page, shows the a block diagram of the PCB design. It is compact and the eight similar modules are connected together on a platform. This method is more advantageous as explained above. It takes more time to be designed but it reduces the space between one display and the neighbouring which makes demonstration optically better.

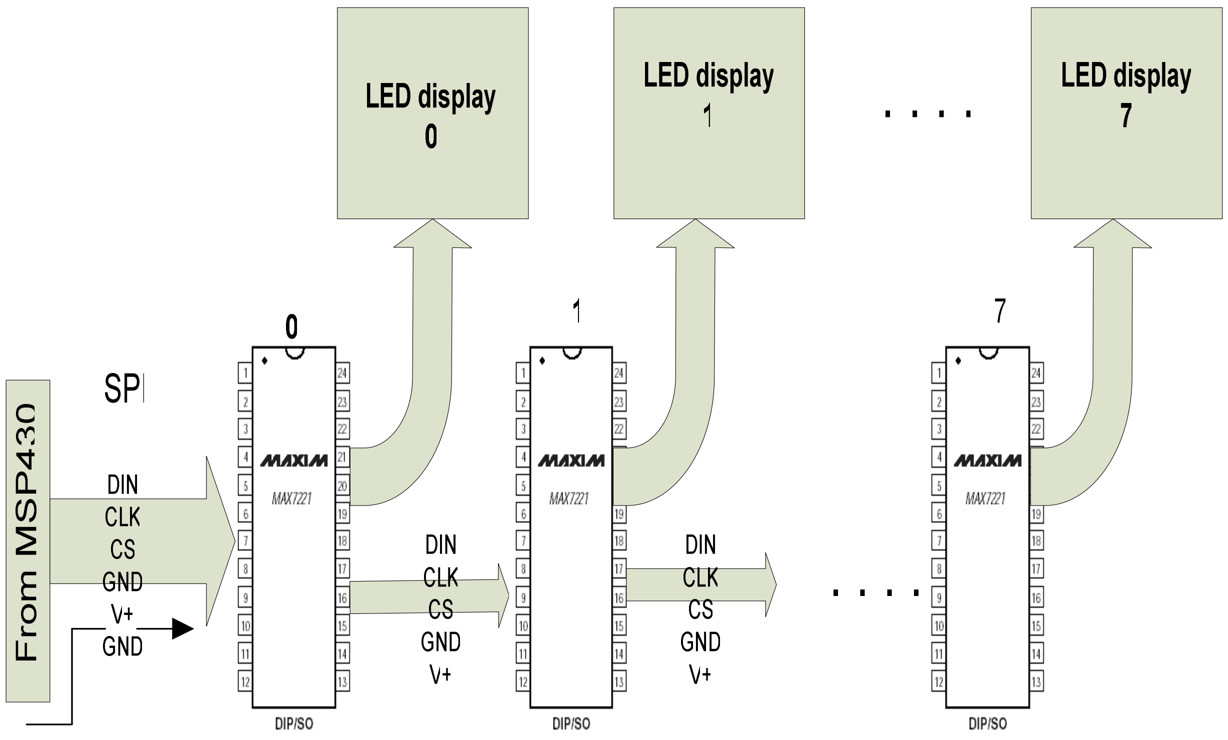
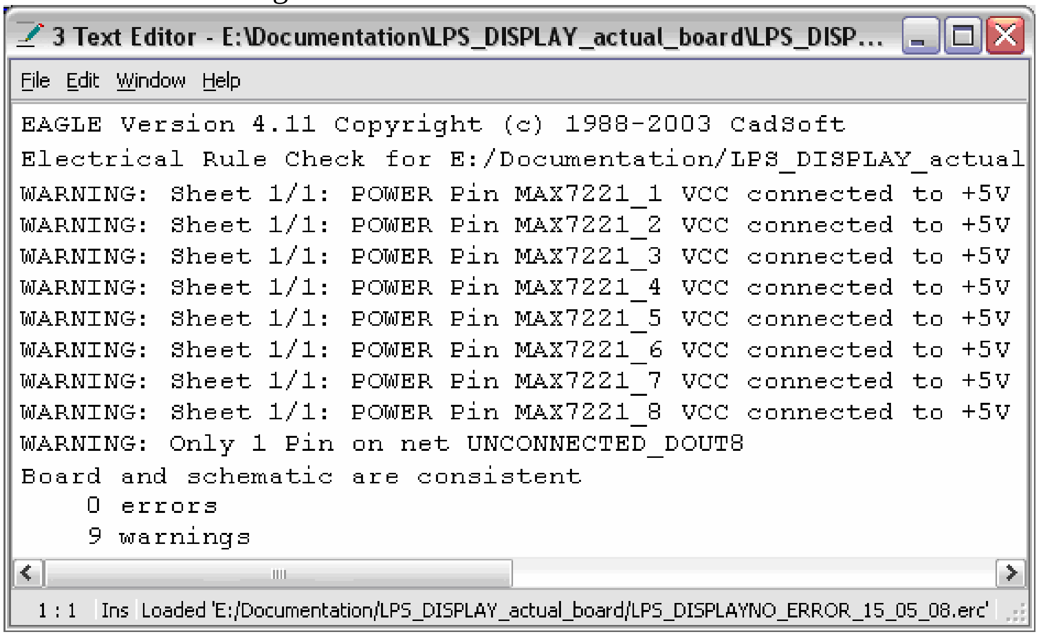


Figure 4.3.4: A block diagram of second design

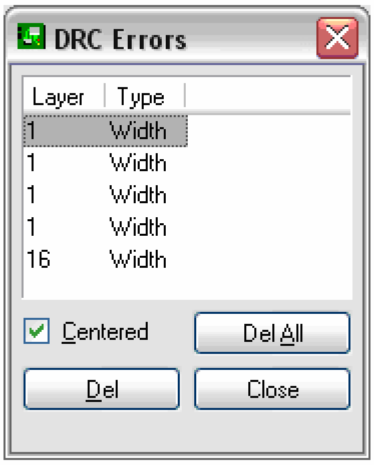
The second design occupies less space and the soldering involved is reduced due to the absence of connectors (16 in number) as in the first method. This also implies the employment of less hardware, thereby making the design cheaper. No complicated structure is needed here for the PCB and LED display to be fixed tight as there are allocations for screws on the board.
Nevertheless, there are a few inconveniences using this method. The Schematics is cumbersome, needing more time for design and control as well as increasing the possibility of having an error that is not detectable with the ERC check. See chapter 8 for the schematics and board of the designs.


3 Text Editor - E:\Documentation\LPS_DISPLAY_actual_board\LPS_DISP...
File Edit Window Help
1 : 1 Ins Loaded 'E:/Documentation/LPS_DISPLAY_actual_board/LPS_DISPLAYNO_ERROR_15_05_08.erc'

```
EAGLE Version 4.11 Copyright (c) 1988-2003 CadSoft
Electrical Rule Check for E:/Documentation/LPS_DISPLAY_actual
WARNING: Sheet 1/1: POWER Pin MAX7221_1 VCC connected to +5V
WARNING: Sheet 1/1: POWER Pin MAX7221_2 VCC connected to +5V
WARNING: Sheet 1/1: POWER Pin MAX7221_3 VCC connected to +5V
WARNING: Sheet 1/1: POWER Pin MAX7221_4 VCC connected to +5V
WARNING: Sheet 1/1: POWER Pin MAX7221_5 VCC connected to +5V
WARNING: Sheet 1/1: POWER Pin MAX7221_6 VCC connected to +5V
WARNING: Sheet 1/1: POWER Pin MAX7221_7 VCC connected to +5V
WARNING: Sheet 1/1: POWER Pin MAX7221_8 VCC connected to +5V
WARNING: Only 1 Pin on net UNCONNECTED_DOUT8
Board and schematic are consistent
    0 errors
    9 warnings
```

Figure 4.3.5: ERC window

After carrying out the electrical rule check of the schematics, there were no errors. Nevertheless, there were a few warnings as seen on Figure 4.2.5above. These warnings are negligible as they are mainly indicating the connection of power supply to nets. The other point is the data out pin of the eighth MAX7221 chip. This pin is open as there are no further connections on it. The consistency of the board and schematics is also confirmed as shown on Figure 4.2.5: ERC window above.



Layer	Type
1	Width
1	Width
1	Width
1	Width
16	Width

Figure 4.3.6: DRC Error window

After designing the PCB, there were a few “errors” from the design rule checks (DRC). These are not important and so are to be ignored. These “errors” are due to the sizes of the name of the board, date, author’s name, version and the name of the laboratory in which the work was done. These notifications could be turned off by increasing the sizes of the named items above. Since it was decided on the given sizes, it is not necessary turning off the notifications.

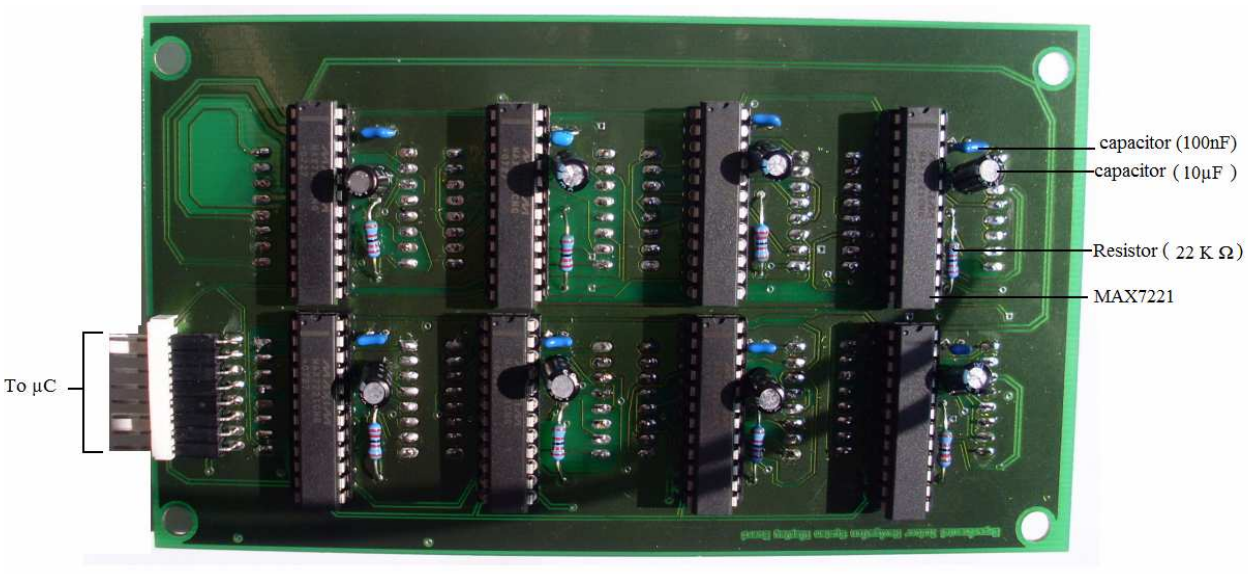


Figure 4.3.7: Produced PCB with electrical components like the IC MAX7221, resistors and capacitors

Figure 4.3.7 is the result of the hardware design. It is compact, double sided and functions as expected. The DIN, CLK, GND and voltage are measured at the various chips and they appear as they are supposed to, and when they are supposed to. These qualify the hardware technically and optically.

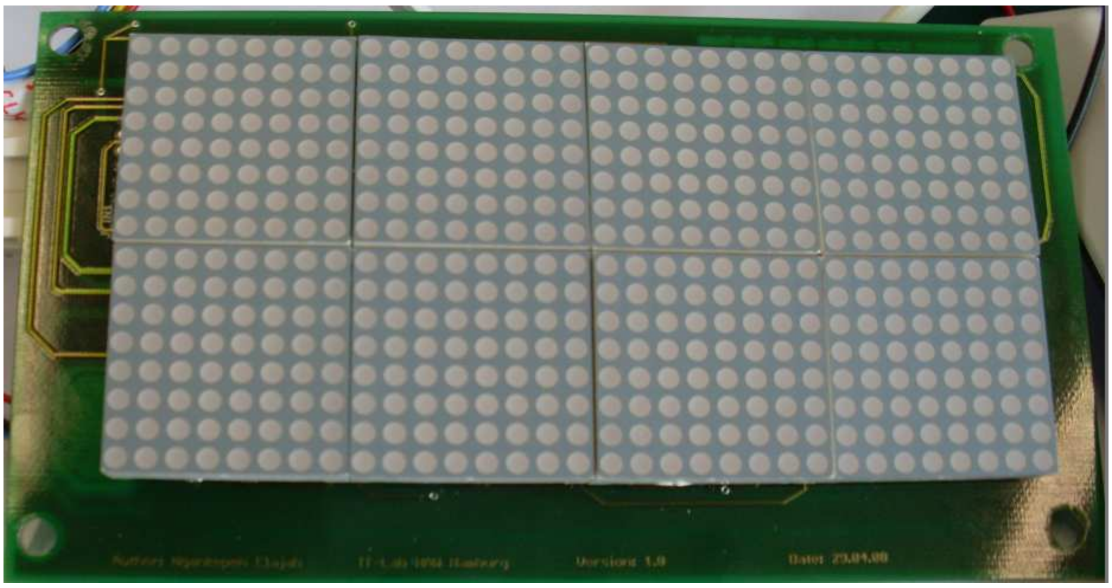


Figure 4.3.8: The full display with 8 LED modules (front view with 512 LEDs)

4.4 -MAX7221 and LED module

The chip MAX7221 and the LEDs are the most important electrical component of the circuit since they play a vital role in the development of the display. As already explained, the MAX7221 are the ICs that make the control of the LEDs realisable. The communication between the microcontroller and the ICs is through USART in SPI mode with the data being latched out with the assistance of the master (MSP430) clock.

4.4.1 - MAX7221

MAX7221 is compact, having serial input/output common-cathode display drivers that interface microprocessor or microcontrollers to 64 individual LEDs. MAX7221 was chosen for this project for couples of reasons. One of the most important reasons, being the availability in small quantity as only 8 of them are necessary to drive 512 LEDs. A further reason is the fact that it permits SPI control and each IC can control up to 64 LEDs. The fact that these ICs could be cascaded to control 512 LEDs also contributed to the decision of choice.
It is compatible with Serial Peripheral Interface (SPI), queued serial peripheral interface (QSPI) and MICROWIRE (µWire), a restricted subset of SPI.
It has slew rate-limited segment drivers to reduce Electromagnetic interference (EMI).

This device include a 150µA shut-down mode, analogue and digital brightness control, a scan limit register that allows the user to display from 1 to 8 digits, and a test mode that forces all LEDs on.
A convenient 4-wire serial interface connects to all common microcontrollers (µCs).
It has a 24-pin dual in-line package (DIP) and SO packages. Figure 4.3.9 shows an overview of a DIP sockets.
A "DIP" is a "dual in-line package“ that has two rows of pins that are in a line. These sockets are usually soldered permanently onto a printed circuit board (PCB). A DIP chip can then be inserted and removed easily as in this case with MAX7221.

The voltage with respect to ground (GND), the operating supply voltage (V+) is between -0.3V and 6V, the DIN, CLK, LOAD and $\overline{CS}$ .All other pins have potential difference from -0.3V to +0.3V. The sink current in DIG 0-7 is 500mA meanwhile the source current of SEG A - G, DP is 100mA.

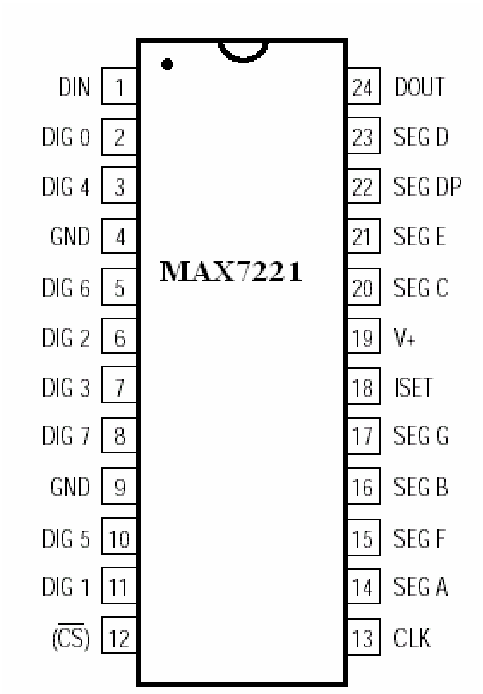


Figure 4.4.0: Pin description of MAX7221

The minimum value for intensity limiting resistor should be 9.53K Ohm. This resistor is either variable or fixed. A fixed resistor was decided for this project as a digital brightness control is possible through software. Its variation influences the brightness of the LEDs.
There are many modes in which the MAX7221 could be driven. There is the serial addressing mode, the shut down mode, the initial power up-mode and the no-Op mode. In the serial addressing mode, CS must be low to clock data in or out. The data is then latched into either the digit or control registers on the rising edge of $\overline{CS}$. $\overline{CS}$ must go high concurrently with or after the 16th rising clock edge, but before the next rising clock edge or data will be lost. Data at DIN is propagated through the shift register and appears at DOUT 16.5 clock cycles later. Data is clocked out on the falling edge of CLK. Data bits are labelled D0–D15. D8–D11 contains the register address. D0–D7 contains the data, and D12–D15 are “don’t care” bits. The first received is D15, the most significant bit (MSB).

Figure 4.4.0 above shows a view of the MAX7221 chip. The pins are described in the table below.

Pin Number	Description
1	DIN - Serial Data In
2	DIG 0 - Digit 0 Drive Line
3	DIG 4 - Digit 4 Drive Line
4	GND - Ground
5	DIG 6 - Digit
6	Drive Line 6 DIG 2 - Digit 2 Drive Line
7	DIG 3 - Digit 3 Drive Line
8	DIG 7 - Digit 7 Drive Line
9	GND - Ground
10	DIG 5 - Digit 5 Drive Line
11	DIG 1 - Digit 1 Drive Line
12	LOAD(CS) - Chip Select
13	CLK - Serial Clock
14	SEG A - Segment A Drive Line
15	SEG F - Segment F Drive Line
16	SEG B - Segment B Drive Line
17	SEG G - Segment G Drive Line
18	ISET - Peak Segment Current
19	V+ - Positive Supply Voltage
20	SEG C - Segment C Drive Line
21	SEG E - Segment E Drive Line
22	SEG DP - Segment Decimal Point Drive Line
23	SEG D - Segment D Drive Line
24	DOUT - Serial Data Out

Table 4.4.0: MAX7221 pin number and description.

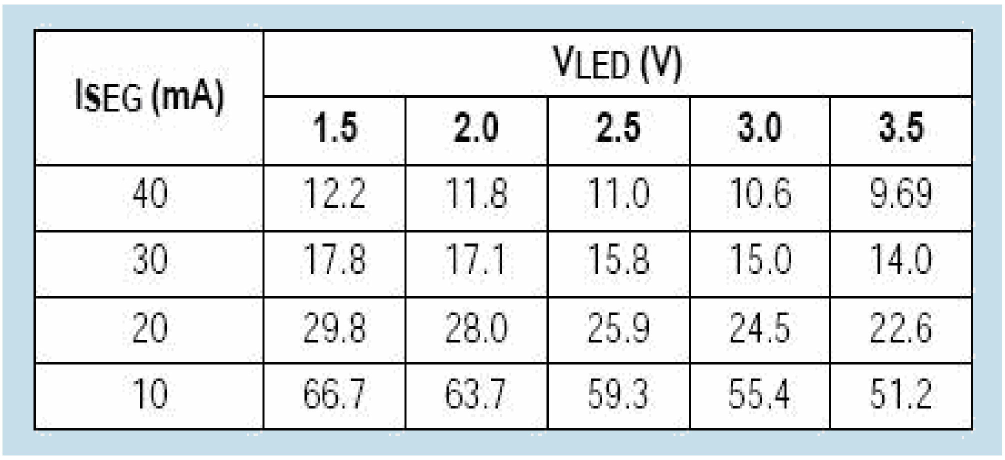
The no-op register is used in cascading MAX7221s. All devices LOAD/ $\overline{CS}$ input are connected together and DOUT connected to DIN on adjacent devices. DOUT is a CMOS logic-level output that easily drives DIN of successively cascaded parts. For example, if four MAX7221s are cascaded, then to write to the fourth chip, the desired 16-bit word is sent, followed by three no-op codes. When $\overline{CS}$ goes high, data is latched (In electronics, a latch is a data storage system used to store information in sequential logic systems. One latch can store one bit of information) in all devices. The first three chips receive no-op commands, and the fourth receives the intended data.

Supply Bypassing and Wiring
To minimize power-supply ripple due to the peak digit driver currents, connect a 10μF electrolytic and a 0.1μF ceramic capacitor between V+ and GND as close to the device as possible. The MAX7221 should be placed in close proximity to the LED display, and connections should be kept as short as possible to minimize the effects of wiring inductance

and electromagnetic interference. Also, both GND pins (pin 4 and 9) must be connected to ground.

Selecting RSET Resistor and Using External Drivers

The current per segment is approximately 100 times the current in ISET. To select RSET, see Table 4.4.1. The MAX7221's maximum recommended segment current is 40mA. For segment current levels above these levels, external digit drivers will be needed. In this application, this driver serves only as controllers for other high-current drivers or transistors. Therefore, to conserve power, R_{SET} = 47kOhm is used when using external current sources as segment drivers. RSET must be selected accordingly.



ISEG (mA)	VLED (V)				
	1.5	2.0	2.5	3.0	3.5
40	12.2	11.8	11.0	10.6	9.69
30	17.8	17.1	15.8	15.0	14.0
20	29.8	28.0	25.9	24.5	22.6
10	66.7	63.7	59.3	55.4	51.2

Table 4.4.1: R_{SET} versus Segment Current and LED Forward Voltage[9]

Cascading drivers

The example in Figure 4.4.1 drives 16 digits using a 3-wire µP interface. If the number of digits is not a multiple of 8, both drivers' scan limits registers are set to the same number, so one display will not appear brighter than the other. For example, if 12 digits are need, 6 digits should be used per display, with both scan-limit registers set for 6 digits so that both displays have a 1/6 duty cycle per digit.

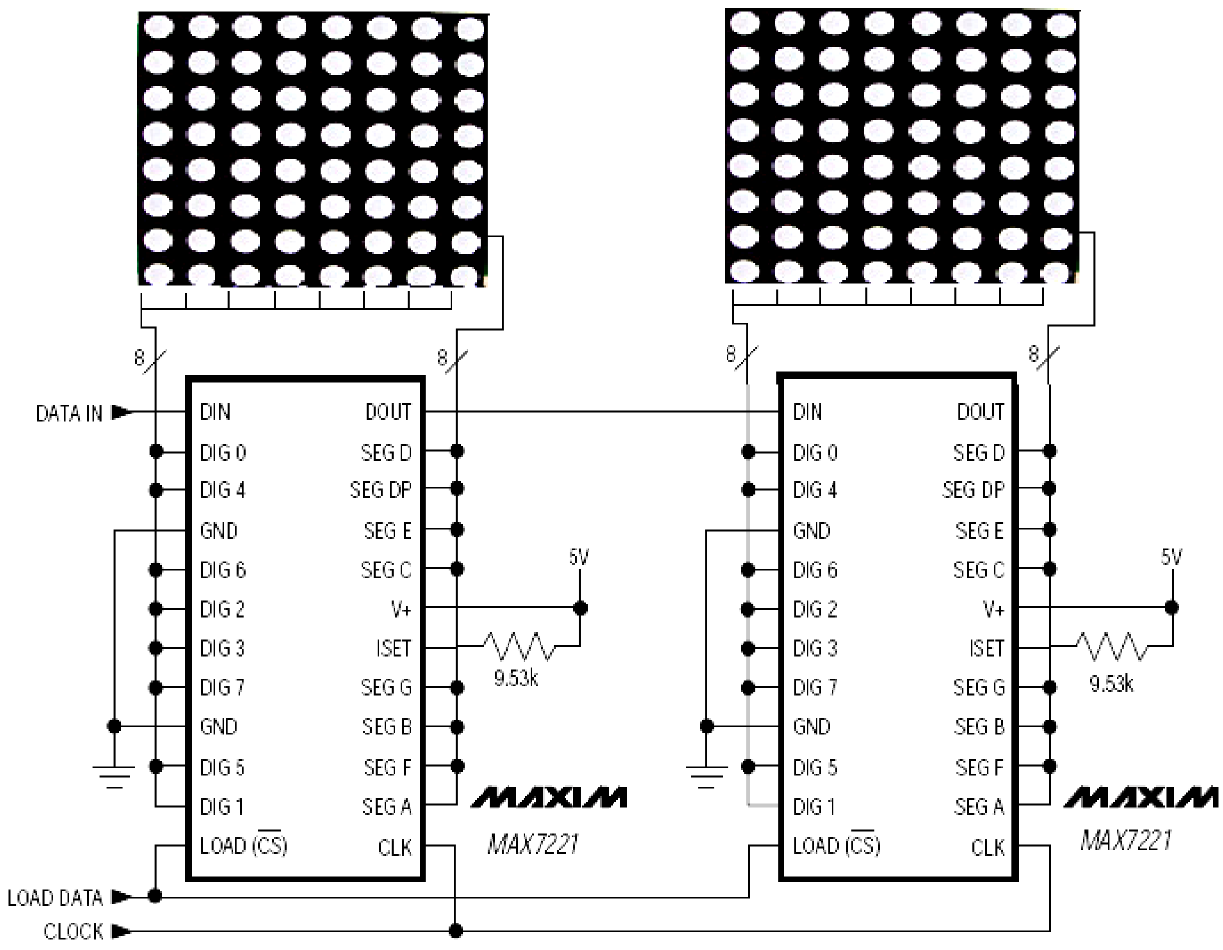


Figure: 4.4.1: cascading MAX7221 (adapted from [9] and [31])

If 11 digits are needed, both scan-limit registers are set for 6 digits and one digit driver left unconnected. If one display is set for for 6 digits and the other for 5 digits, the second display will appear brighter because its duty cycle per digit will be 1/5 while that of the first display will be 1/6. Refer to the No-Op Register section for additional information.
Figure 4.4.1 shows two MAX/221 IC cascaded together. In this project, eight of them have been cascaded together, with the DOUT of a MAX7221 being the DIN of the next.

Computing Power Dissipation

The power dissipation is a factor worth analysing, so as to know the limits of current for the system. The upper limit for power dissipation (PD) for theMAX7221 is determined from the following equation:

PD = (V + x 8mA) + (V+ - VLED)*(DUTY x ISEG x N) where:
V+ = supply voltage
DUTY = duty cycle set by intensity register
N = number of segments driven (worst case is 8)
VLED = LED forward voltage
ISEG = segment current set by RSET

Dissipation Example:
ISEG = 40mA, N = 8, DUTY = 31/32, VLED = 1.8V at 40mA, V+ = 5.25V

PD = 5.25V (8mA) + (5.25V - 1.8V) (31/32 x40mA x 8) = 1.11W
Thus, for a CERDIP package (qJA = +60°C/W [9]), the maximum allowed ambient temperature TA is given by:

TJ (MAX) = TA + PD *qJA + 150°C (4.3)
= TA +1.11W x 60°C/W (4.4)
From the above equations, TA = +83.4°C.

4.4.2 - LED Display (Model No: ELM-1883SRWA)

4.4.2.1 - General characteristics of LEDs

4.4.2.1.1 -A brief History

Light Emitting Diodes (LEDs) are expected to be used in a variety of new applications as a next-generation lighting source. The demands on electronic products go towards cheaper and easier to handle products that show the full, or even better optical detectors performance of expensive products. In the area of optical detectors, this demand leads to integrated photodiodes and receiver on one chip. [2]
Commercial research into LED technology started in 1962 at Bell Labs, Hewlett-Packard, IBM, Monsanto and RCA [11]. Work on gallium arsenide phosphate (GaAsP) led HP and Monsanto to introduce the first commercial 655nm red LEDs in 1969. In 1971, HP released the 5300A 500MHz portable frequency counter using GaAsP LED display. The Commission Internationale de l'Eclairage (CIE) formalized standards for measuring light and the response of the human eye to light in the 1930s. This commission defined the primary colours including their wavelength. Radiant light intensity (wavelengths) is measured in lumens meanwhile luminous intensity is measured in candelas (cd). The lumen definition states that 683 lumens of light are provided by 1 watt of monochromatic radiation at a wavelength of 555nm. The mechanical construction of the LED lamp determines the radiated light pattern. A narrow radiated pattern will appear very bright.
LEDs are processed in wafer forms that are similar to silicon-integrated circuits and broken out into dice.

4.4.2.1.2 - The electrical and optical Characteristics of LEDs

LEDs are usually small, have a long life with mean time between failures (MTBF) of about 100,000 to over 1,000,000 hours for a continuous operation. Their behaviours are similar to other semiconductor diodes. The forward voltage in LEDs is higher and differs with different materials used for different colours. The forward voltage increases with current and decreases with temperature. The voltage decreases with temperature with a gradient of about 2mV/°C.
The optical comportment of LEDs varies remarkably with temperature. The quantity of light emitted by the LED lamp falls as the junction temperature rises. This is due to a rise in the recombination of holes and mainly because the energy gap of the semiconductor varies with

temperature. LEDs of different colours have different wavelengths. Table 3.3.2.1 shows this variation for the three primary colours.

Colour name	Wavelength in nanometres
Red	700
Green	5461.1
Blue	435.8

Table 4.3.2.1: Primary colours and their wavelengths

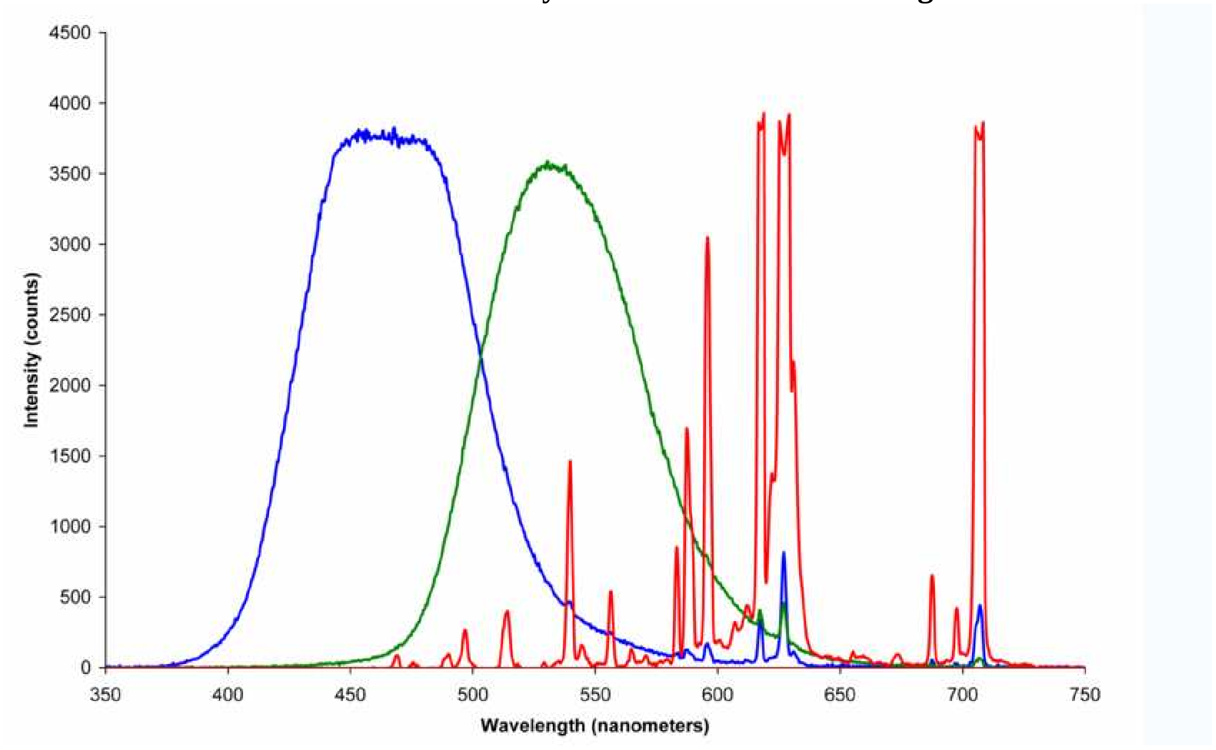


Figure 4.3.2.1: Spectra of individual primary colours [22]

The behaviour of the three primary colours as far as intensity and wavelength (lambda: λ) is concerned is demonstrated in Figure 3.3.2.1 above. It is visible that at lower values of λ, the blue colour attains its maximum intensity. This is followed by the green and then the red colour. The explanation is simply the fact that at values of λ around their various wavelengths, they attain their maximum intensity. Table 3.3.2.1 and Figure 3.3.2.1 explain this behaviour closely. These explanations help understand some of the characteristics of LED and their colours so as to judiciously handle them in hardware and software combination.
A simple electrical circuit diagram of an LED basically consists of a DC source, an LED and a resistor. See the Figure 3.3.2.1.2 below for further clarifications.

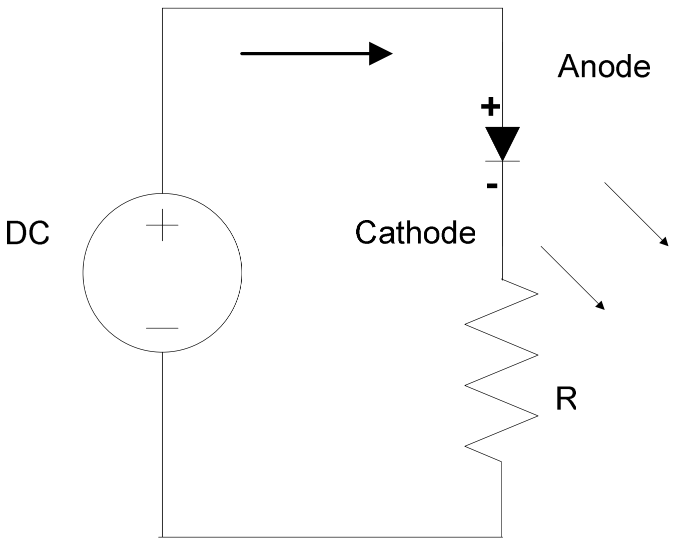


Figure 4.3.2.1.2: LED circuit

The correct resistance of the resistor R could be calculated using the formula below:

$$\mathrm{R} = \frac{P - V_{LED}}{I_{LEDrating}} \qquad (4.3.2.1)$$

Where:

R is the resistor in Ohms (Ω), P is the Power supply voltage (such as a 5 volt battery) V_{LED} is the voltage drop across the LED (typically about 1.7 - 3.3 volts; this varies with the colour of the LED), $I_{LEDrating}$ is the LED current rating from the manufacturer of the LED (usually given in milliamperes).

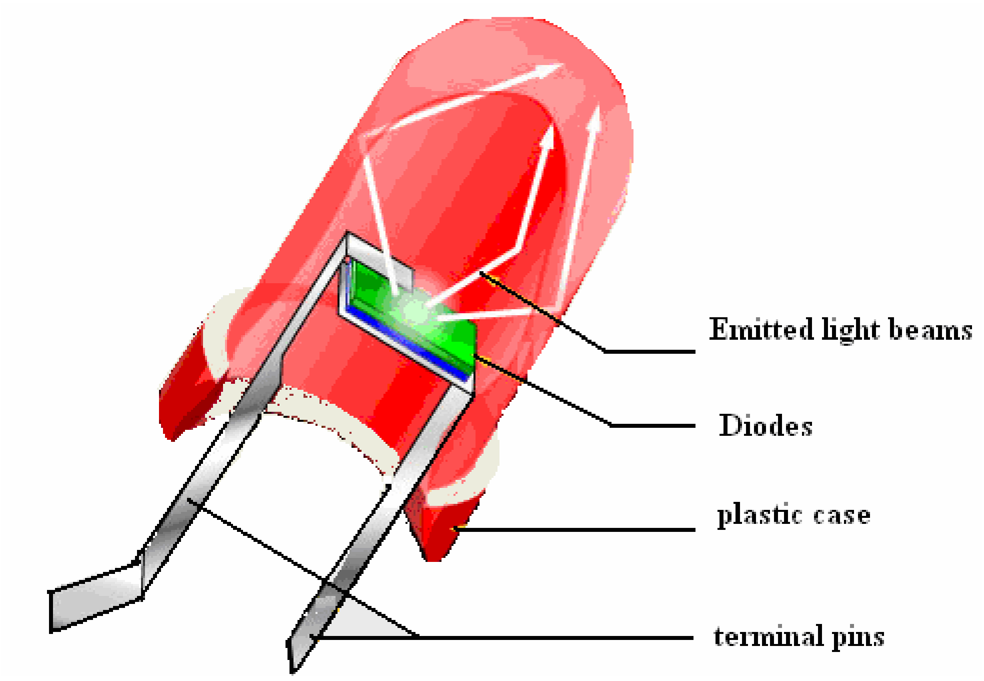


Figure 4.3.2.1.3: The inner view of a single LED (adapted from [32])

The LED used in this project is an 8*8 Dot Matrix Display with large emitting dots. This series of display have a large emitting area (3.0mm diameter) and LED sources configured in a 64 dots 8*8 matrix array. The device is made up of white surface and grey dots. From the centre of an LED to the next is 4mm, making the total of 28mm as the width or height of the device from the centre of the first LED to the centre of the last (8th) LED. The total width is 28mm + 4mm = 32mm. From one pin of the LED to the next is 2.54mm
It is a low power and high brightness LED, making it suitable to its use as a display for a battery supplied device. The emitted colour is red. The intensity is either digitally controlled using the intensity register or through a variable resistor 47KΩ. The maximum recommended segment current is 40mA. [9]

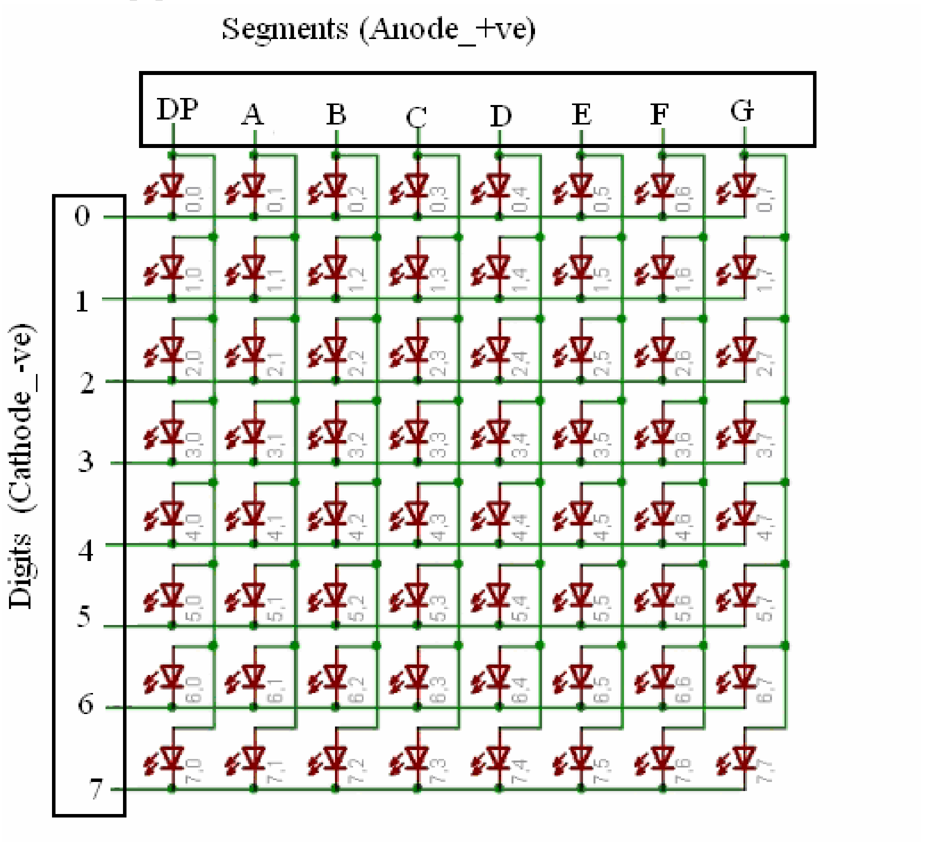


Figure 4.3.2.1.4: 8x8 LED matrix (inside view)

An LED matrix display as shown on Figure 4.3.2.1.4 is made up of 64 LEDs and they have a common cathode. The address of each LED is also shown. An example of an address is (0, 6). The 0 stands for the row and the 6 stands for column. The anode is connected on the segments from A to G including the decimal point DP meanwhile the cathode is connected to the digits (DIG0 to DIG7) as seen in Figure 4.3.2.1.4 above.

4.4.2.1.3 - Reliability and Lumen Maintenance

This sub section outlines an important point in choosing the LED as a display. Lumen maintenance is simply the amount of light emitted from a source at any given time relative to the light output when the source was first measured. It is expressed as a percentage. The effects of lumen depreciation are noticed for example when changing an old light bulb with a new one. This steady decline over time is known as lumen maintenance . The materials inside the bulb will continue to deteriorate until finally the bulb will no longer emit any light. This is

known as the bulb's mortality. With conventional light sources, the bulb usually fails before our eyes notice the change in lumen maintenance.
LEDs also experience lumen depreciation but it happens over a much longer period of time, usually tens of thousands of hours. Compared to conventional light sources, one notices very little degradation of light with LUXEON LEDs.
Not all LEDs deliver the same lumen maintenance. An LED is a complex package of materials that must all work together to deliver long lifetimes. Everything from the design of the chip, thermal management, optics material, phosphors and even the assembly of the entire package will affect lumen maintenance. Some LEDs demonstrate very rapid depreciation but Philips Lumileds technical advances in all of these areas has led to the industry's longest lived LEDs — LUXEON. [11]

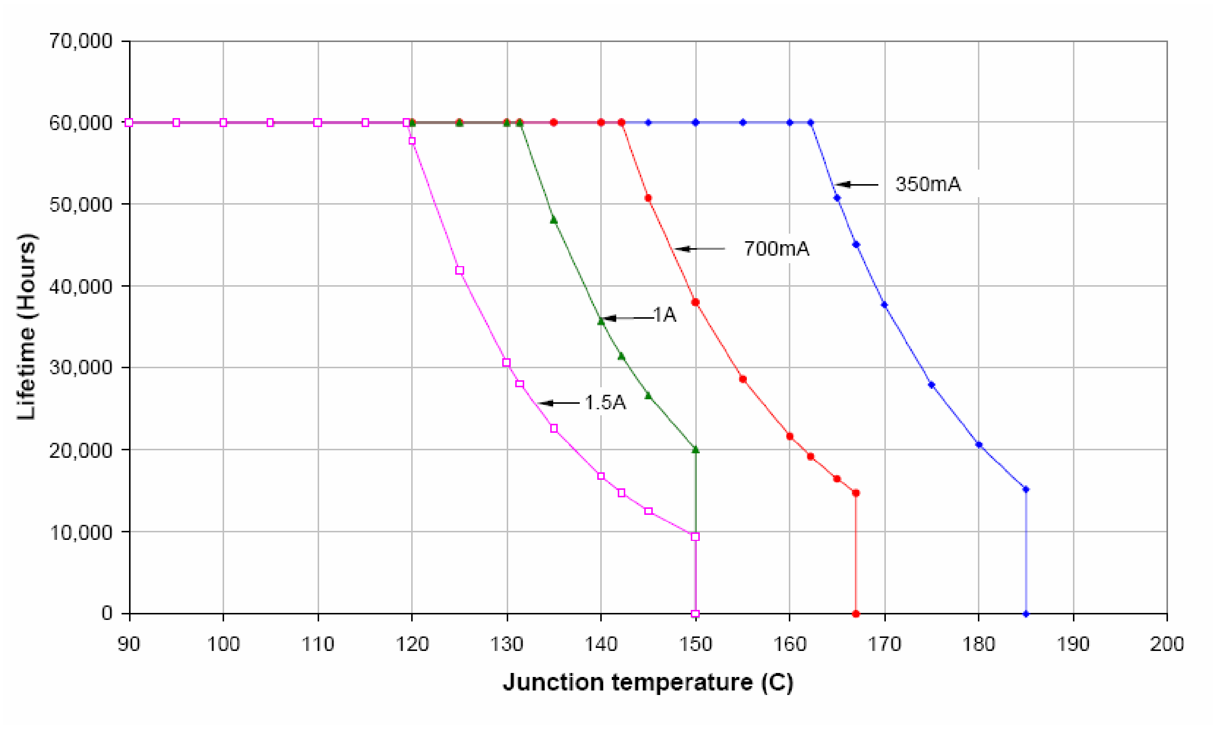


Figure 4.3.2.1.1: Lifetime data across current and temperature variables for a K2 LED [11]

The junction temperature of LEDs is a very important factor in considering an LED. The curves in Figure 4.3.2.1.1 show a variation of junction temperature with life time of the LEDs in hours for four different currents (1.5A, 1A, 700mA and 350mA). As shown in the curves, an increase of current will lead to an increase of the junction temperature. This helps in giving a guide line to maintaining the current at a minimum level to avoid excessive heating.
LED lifetime could be predicted using the Weibull distribution function.
The Weibull distribution function can be written as:

$$f(x;k,\lambda)=\frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1}e^{-(\frac{x}{\lambda})^k} \quad [11] \qquad \textbf{(4.3.2.1)}$$

where x, k and λ are derived from experimental data based on collected data, such as recorded lumen maintenance of power LEDs undergoing lifetime testing, values for x, k and λ can be calculated. The Weibull distribution function is then used to extrapolate the recorded data to predict behaviour over a period longer than that measured. The application of the Weibull function acknowledges that the size of the recorded data set, influences the accuracy of the calculated values x, k and λ. A larger body of experimental data produces values that more closely resemble the behaviour of the system. The ideal LED driving circuit is optimal when the LED characteristics listed are taken into consideration. LEDs are electronic devices that emit light by the delivery of current, so appropriate power sources are required to drive them.

Parameter	Symbol	Rating	Unit
Reverse Voltage	Vr	5	V
Forward Current	If	40	mA
Operating Temperature	Topr	-40 to +85	℃
Storage Temperature	Tstg	-40 to +100	℃
Soldering Temperature	Tsol	260 ± 5	℃
Power Dissipation	Pd	110	mW
Peak Forward Current(Duty 1/10 @ 1KHZ)	If(Peak)	180	mA

Table: **4.3.2.1.1.1: Absolute maximum ratings at 25°C (Ta)**

Table: 4.3.2.1.1.1 gives the ratings for the reverse voltage, forward current, operating temperature, storage temperature, soldering temperature, power dissipation and pick forward current.

Parameter	Symbol	MIN.	TYP.	MAX.	Unit	Condition
Luminous Intensity	Iv	5.6	8.0	----	mcd	If=10mA
Peak Wavelength	λ p	----	660	----	nm	If=20mA
Dominant Wavelength	λ d	----	643	----	nm	If=20mA
Spectrum Rediation Bandwidth	△ λ	----	20	----	nm	If=20mA
Forward Voltage	Vf	1.5	1.7	2.4	V	If=20mA
Reverse Current	Ir	----	----	10	μ A	Vr = 5V

Table: **4.3.2.1.1.2: Electronic optical characteristics of LED used**

No	Item	Test conditions	Test Hours/Cycle	Sample size
1	Solder heat	TEMP:260°C ±5°C	5SEC	76 PCS
2	Temperature cycle	H:+85°C 30min 5min L:-55°C 30min	50 CYCLES	76 PCS
3	Thermal shock	H: +100°C 10min L:-10°C 5min	50 CYCLES	76 PCS
4	High temperature storage	TEMP:100°C	1000 CYCLES	76 PCS
5	Low temperature storage	TEMP:-55°C	1000 CYCLES	76 PCS
6	DC operating life	If = 20mA	1000 CYCLES	76 PCS
7	High temperature / High humidity	85°C/85%RH	1000CYCLES	76 PCS

Table: 4.3.2.1.1.3: Reliability test item and condition for the LEDs used (ELM-1883SRWA) [29]

Table 4.3.2.1.1.3 gives brief extreme conditions under which some LEDs have been tested. These limits are not to be exceeded, so as to avoid abnormal behaviour of the LEDs.

The LEDs (ELM-**1883SRWA**) have some typical electro-optical properties that need to be taken in to consideration to avoid damages and faulty results. The curves below give more details to these properties.

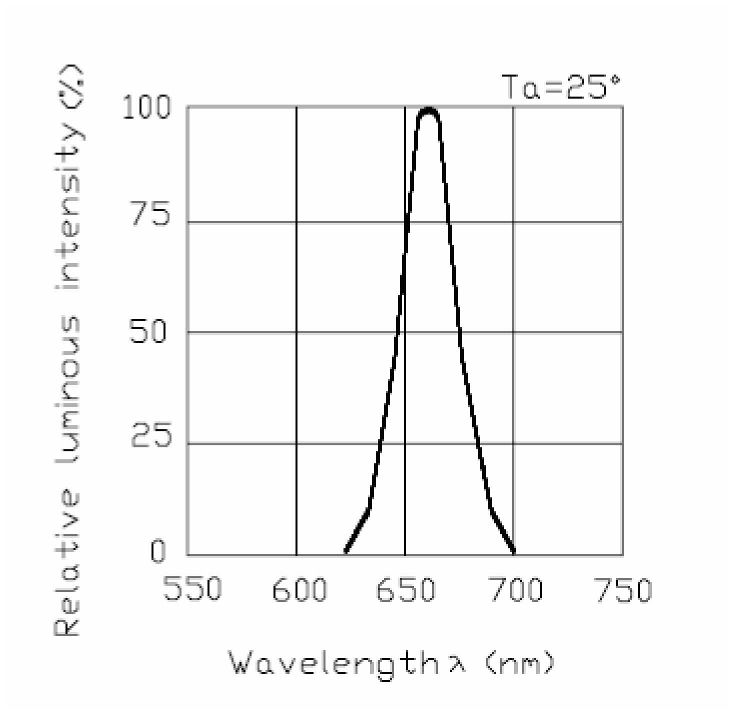


Figure 4.3.2.1.1.0: Spectrum distribution

The luminous intensity has a short range of wavelength (~620-700nm) over which the percentage is above zero.

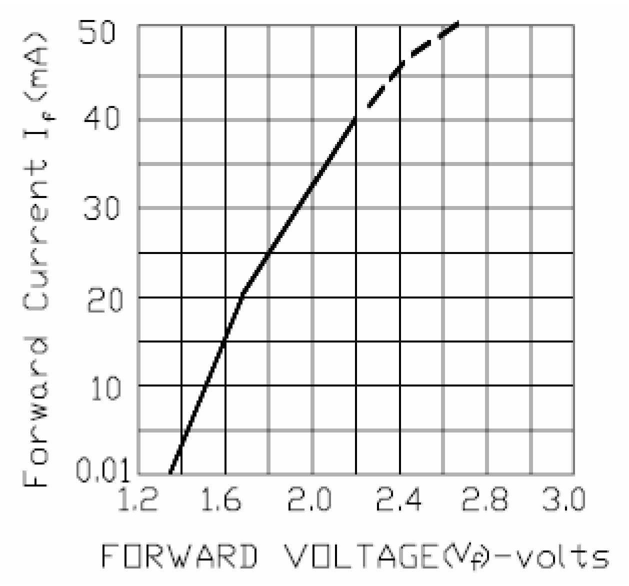


Figure 4.3.2.1.1.1: Forward voltage versus forward current.

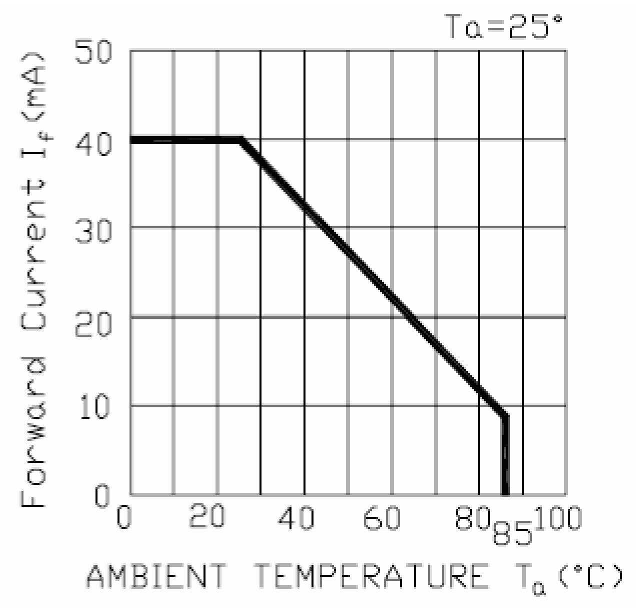


Figure 4.3.2.1.1.2: Forward current – ambient temperature curve.

The variation of forward current with the ambient temperature is represented in Figure 4.3.2.1.1.2. The forward current remains stable from 0°C to room temperature and then begins to decrease with further increase of ambient temperature. At 85°C, the current falls to zero, implying these LEDs could not be controlled at such temperatures. This property is very important in limiting as temperatures above 85 could damage the device.

4.4.3 – Advantages and Disadvantages of using LEDs

4.4.3.1 - Advantages of using LEDs

- LEDs produce more light per watt than incandescent bulbs; this is useful in battery powered or energy-saving devices. LEDs can emit light of an intended colour without the use of colour filters that traditional lighting methods require. This is more efficient and can lower initial costs.
- The solid package of the LED can be designed to focus its light. Incandescent and fluorescent sources often require an external reflector to collect light and direct it in a usable manner.
- When used in applications where dimming is required, LEDs do not change their color tint as the current passing through them is lowered, unlike incandescent lamps, which turn yellow.
- LEDs are ideal for use in applications that are subject to frequent on-off cycling, unlike fluorescent lamps that burn out more quickly when cycled frequently, or HID lamps that require a long time before restarting.
- LEDs, being solid state components, are difficult to damage with external shock. Fluorescent and incandescent bulbs are easily broken if dropped on the ground.
- LEDs can have a relatively long useful life. One report estimates 35,000 to 50,000 hours of useful life, though time to complete failure may be longer. [11] Fluorescent tubes typically are rated at about 30,000 hours, and incandescent light bulbs at 1,000–2,000 hours. [11]
- LEDs mostly fail by dimming over time, rather than the abrupt burn-out of incandescent bulbs .

- LEDs light up very quickly. A typical red indicator LED will achieve full brightness in microseconds; Philips Lumileds technical datasheet DS23 for the Luxeon Star states "less than 100ns." LEDs used in communications devices can have even faster response times.
- LEDs can be very small and are easily populated onto printed circuit boards.
- LEDs do not contain mercury, unlike compact fluorescent lamps.
- Low voltage supply means simple installation.
- Low heat dissipation. That means it does not influence measured object due to heat dissipation.
- Unlike most other illumination sources the output of LED lights are very close to linear. [12]

4.4.3.2 - Disadvantages of using LEDs

Though there are more advantages of using the LED matrix as a display, there are nevertheless a few disadvantages. Some of these are as include:

- LEDs cannot be used in applications that need a sharply directive and collimated beam of light. LEDs are not capable of providing directivity below a few degrees. In such cases LASERs (or LED lasers) may be a better option.
- LEDs are currently more expensive, price per lumen, on an initial capital cost basis, than more conventional lighting technologies.
- The performance of LEDs depends much on the ambient temperature of the operating environment. Overheating may lead to overheating of the LED package, consequently leading to device malfunctioning or even failure. Proper heat-sinking is required to maintain long life.

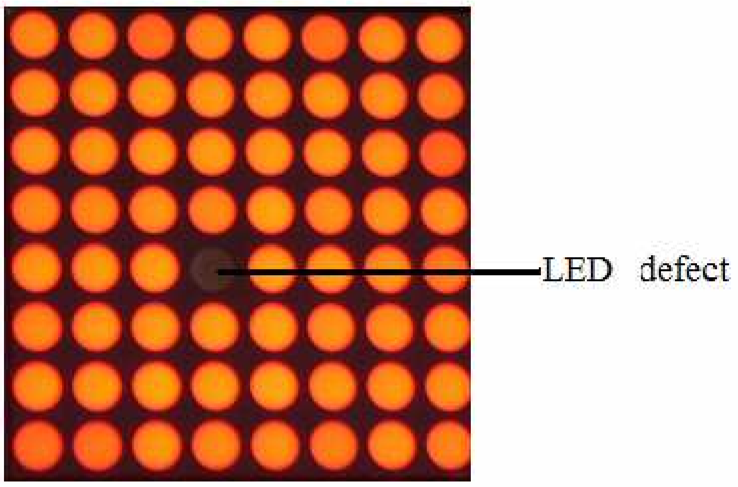


Figure 4.3.2.1.3.1: LED display showing a defective LED in the second PCB

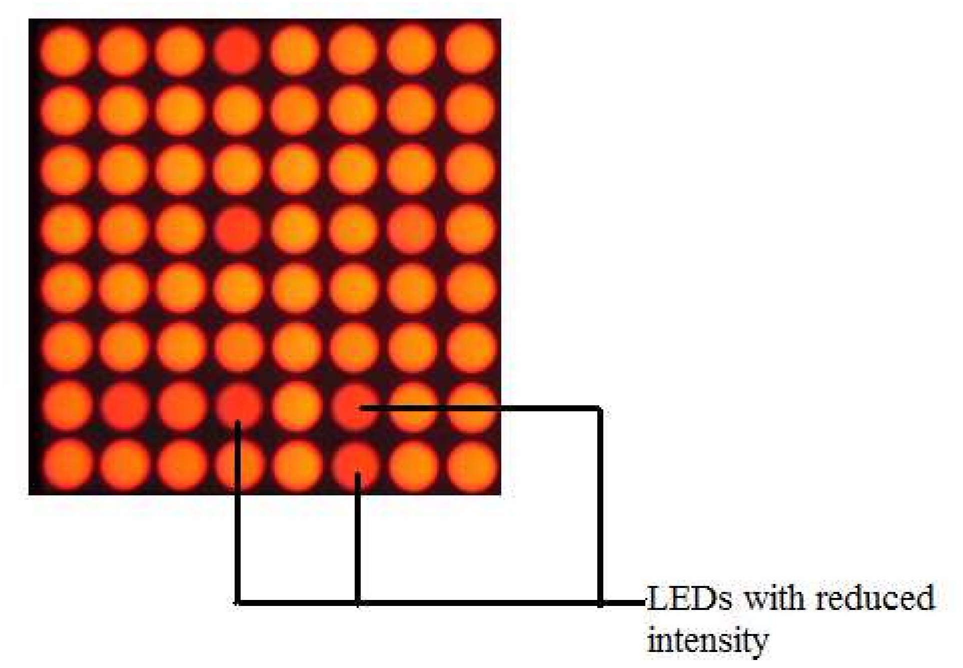


Figure 4.3.2.1.3.2: LED display showing some disadvantages of LEDs as a display

Though the intensity of the LED displays has been set to the same value in the intensity register, the intensity of some LEDs is weaker.

4.5 - MSP430 JTAG Connector

JTAG stands for Joint Test Action Group. [13] It is an IEEE standard for boundary scan technology. MSP430-JTAG is used for programming and flashing emulation with MSP430 microprocessors.

4.5.1 – Features of JTAG

MSP430-JTAG connects to LPT parallel port. This permits writing and debugging code in C language for all MSP430 microcontrollers. MSP430-JTAG has the advantage that it does not need external power source since MSP430 microcontrollers require only 3-5 mA while programming, and all necessary power supply is taken from the LPT port.

Figure 4.4.1: JTAG connector

4.5.1.1 - JTAG interface

The JTAG connector is 2x7 pin with 0.1" step and TI recommended JTAG layout. The PIN.1 is marked with square pad on bottom and arrow on top.

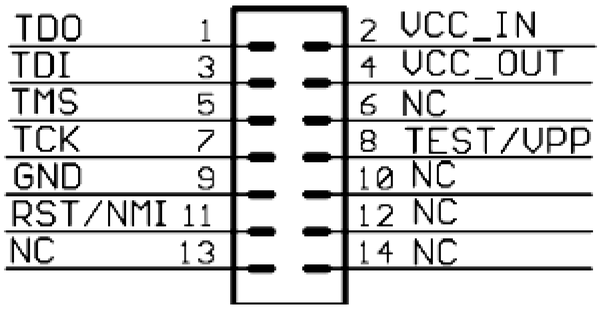


Figure 4.4.2: MSP430-JTAG interface (top view)

There are 14 pins for the JTAG interface of which some are not connected. Those that are not connected are labelled NC. TMS - Test Mode Select (input from controller) (pull up required to force entry into reset state in event of a bad connection on TMS).
TCK is the Test Clock (input from controller).
TDI - Test Data In (input from controller).
TDO - Test Data Out (output to controller or next chip in chain).
The above, function hand in hand, making it possible to write and debug the code in MSP430.

4.6 - Recommended Standard 232 (RS232) connections

RS-232 (Recommended Standard 232) is a standard for serial binary data signals connecting between a DTE (Data Terminal Equipment) and a *DCE* (Data Circuit-terminating Equipment). It is commonly used in computer serial ports as well as in microprocessors and microcontrollers. The RS-232 standard defines the voltage levels that correspond to logical one and logical zero levels. Valid signals are plus or minus 3 to 15 volts. The range near zero volts is not a valid RS-232 level; logic one is defined as a negative voltage, the signal condition is called marking, and has the functional significance of OFF. Logic zero is positive, the signal condition is spacing, and has the function ON.
RS232 Standard needs a minimum interface of 3 wires
– 1 Signal from TxDA to RxDB
– 1 Signal to RxDA from TxDB
– 1 Wire GND = common ground signal [5]
Most serial communications structures send the data bits within each byte such that LSB (Least Significant Bit) is first. [5] This is equally known as "little endian" transmissions. It is also possible, but very rare to have a transmission in which the MSB (Most Significant Bit) in a Byte is transferred first. This is also called “big endian”.

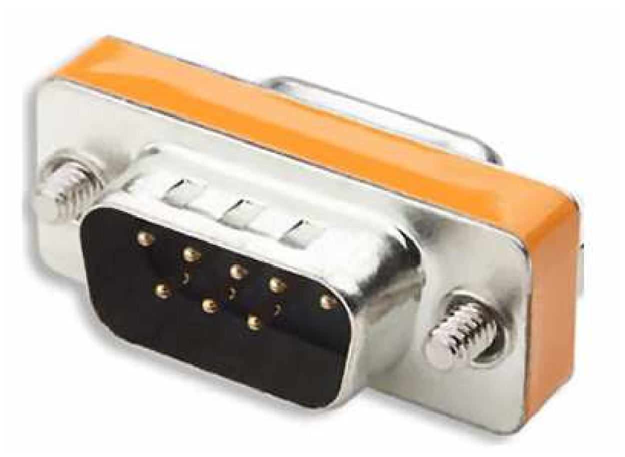


Figure 4.4.3: Male RS232 connector

The male RS232, likewise the female has 9 pins though extended out to fit with the 9 holes of the female. Figure 4.4.3 shows the male while Figure 4.4.4 the female.

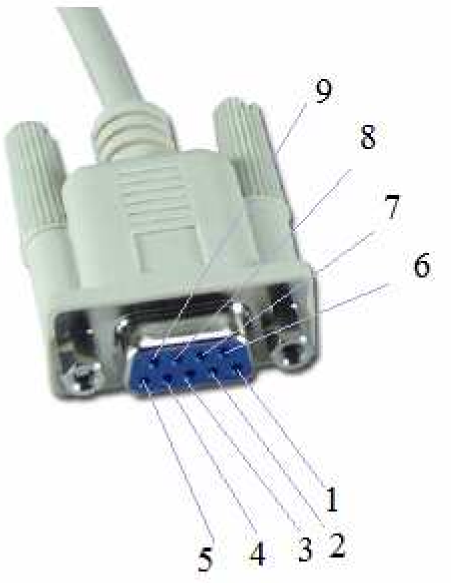


Figure 4.4.4: Female RS232 connector

The RS232 connector is used in the UART communication between two MSP430. The connector is shown in Figure 4.4.3 and 4.4.4. The holes, numbered from 1 to 9 have their various functions. These functions are listed in Table 4.4.1 below. Since the female side is to be connected on the MSP430F169 STK, it will be necessary using an RS323 that both ends are female or else they should be adapted to suit this purpose. During the project, for test purpose, the male-female connector was used as communication (RS232) was between a microcontroller, needing a female and the PC needing a male.

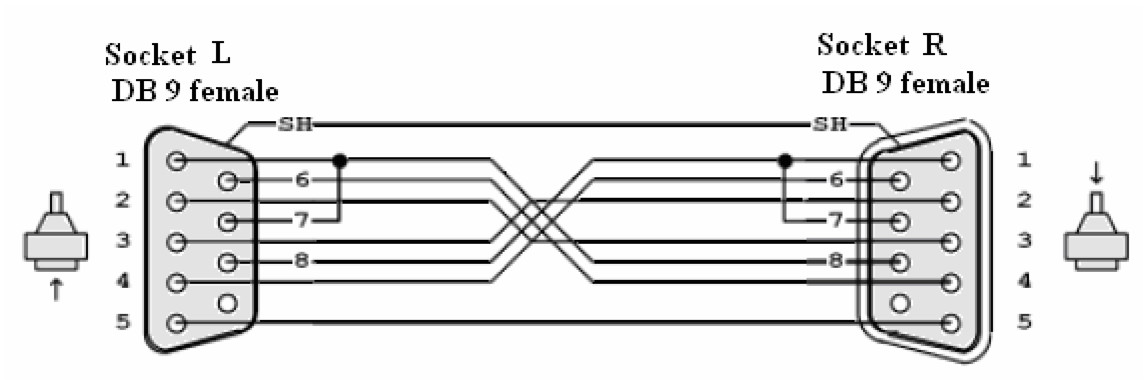


Figure 4.4.5: "Null Modem cable" for two controllers [5]

As shown in Figure 4.4.5, there is crossover of TxD to RxD and RxD to TxD, since two females are connecting. There is no crossover in the case of a male-female.
Some abbreviations used in RS232 interface between two MSP430 are explained below.
Transmit Data – Home MSP430F169 serial data output stream to foreign MSP430F169
Receive Data – Home MSP430 serial data input stream from foreign MSP430F169
GND - Signal ground
Carrier Detect - tells foreign MSP430F169 that MSP430F169 has good connection
Clear To Send - MSP430F169 is ready to receive data from foreign MSP430F169
Data Set Ready – indicates the readiness for data reception
Data Terminal Ready - tells modem that PC is ready
Request To Send – tells home MSPF169 that foreign MSP430F169 wants to send data

Signal Type	**Abbreviation**	**Direction**	**DE-9**
Common Ground	G	-	5
Transmitted Data	TxD	out	3
Received Data	RxD	in	2
Data Transmit Ready	DTR	out	4
Data Set Ready	DSR	in	6
Request To Send	RTS	out	7
Clear To Send	CTS	in	8
Carrier Detection	DCD	in	1
Ring Indicator	RI	in	9

Table 4.4.5: RS232 - 9 pins signal type and direction

Table 4.4.1 above gives an overview of an RS232 signal types and the directions of signals.

Chapter 5: Software structuring and development

After selecting the desired hardware, an appropriate software design is necessary, though the choice of the hardware also involve the viability of the software design.
Hardware and software co-design is not new but has been gaining more and more attention in the past years due to the search for a better method of controlling, regulation and generally automating systems. A major problem in the architectural development of systems is finding a pattern of the system's function as far as the components is concerned, which are regarded as the systems resource. The overall cost of such a system like that in this project does not depend solely on the hardware, since the degree of complexity of the software design is dependent on the hardware available or obtained. An example is designing software for an LCD display and another for an LED display. A given system will as well deliver higher performance when the hardware design is tuned to the software application likewise vice versa. Micro-controllers with the aid of sensors and actuators allow the system to communicate with the environment often called reactive systems. They react to the environment by executing functions in predefined time windows and events. An example of event reaction is the "unavoidable" use of interrupts. When certain events occur, an interrupt system can signal the processor to suspend processing the current instruction sequence and to begin an interrupt service routine (ISR). The ISR will perform any processing required based on the source of the interrupt before returning to the original instruction sequence.
The software has been developed in a modular form. This was not the first design since the complexity of the software part could not be identified at the beginning of the project.
A few tools were used in realising the software programming and it is worth mentioning the quality and reason of choice of compiler, and a few other software tools.

5.1 - The GCC toolchain for the Texas Instruments MSP430 microcontrollers

5.1.1 - A Brief History of GCC

The **GNU Compiler Collection** (usually shortened to **GCC**) is a set of compilers produced for various programming languages by the GNU Project. GCC is a key component of the GNU toolchain. As well as being the official compiler of the GNU system, GCC has been adopted as the standard compiler by most other modern Unix-like computer operating systems, including Linux, the BSD family and Mac OS X. GCC has been ported to a wide variety of computer architectures, and is widely deployed as a tool in commercial, proprietary and closed source software development environments. In addition to the processors used in personal computers, it also supports microcontrollers, DSPs and 64-bit CPUs. GCC is also used in popular embedded platforms like MSP430, Symbian, Playstation and Sega Dreamcast. GCC has a modular design, allowing support for new languages and architectures to be added. Adding a new language front-end to GCC enables the use of that language on any architecture, provided that the necessary run-time facilities (such as libraries) are available. Similarly, adding support for a new architecture makes it available to all languages.

The original author of the GNU C Compiler (GCC) is Richard Stallman, the founder of the GNU Project. The GNU Project was started in 1984 to create a complete Unix-like operating system as free software, in order to promote freedom and cooperation among computer users and programmers. Every Unix-like operating system needs a C compiler, and as there were no free compilers in existence at that time, the GNU Project had to develop one from scratch. The work was funded by donations from individuals and companies to the Free Software Foundation, a non-profit organization set up to support the work of the GNU Project.

The first release of GCC was made in 1987. This was a significant breakthrough, being the first portable ANSI C optimizing compiler released as free software. Since that time GCC has become one of the most important tools in the development of free software.

A major revision of the compiler came with the 2.0 series in 1992, which added the ability to compile C++. In 1997 an experimental branch of the compiler (EGCS) was created, to improve optimization and C++ support. Following this work, EGCS was adopted as the new main-line of GCC development, and these features became widely available in the 3.0 release of GCC in 2001. Over time GCC has been extended to support many additional languages, including FORTRAN, ADA, Java and Objective-C. The acronym GCC is now used to refer to the "GNU Compiler Collection". The C language could be used with good efficiency in microcontrollers. [23]

The following packages are included in the MSPGCC:

- GNU Debugger (GDB)

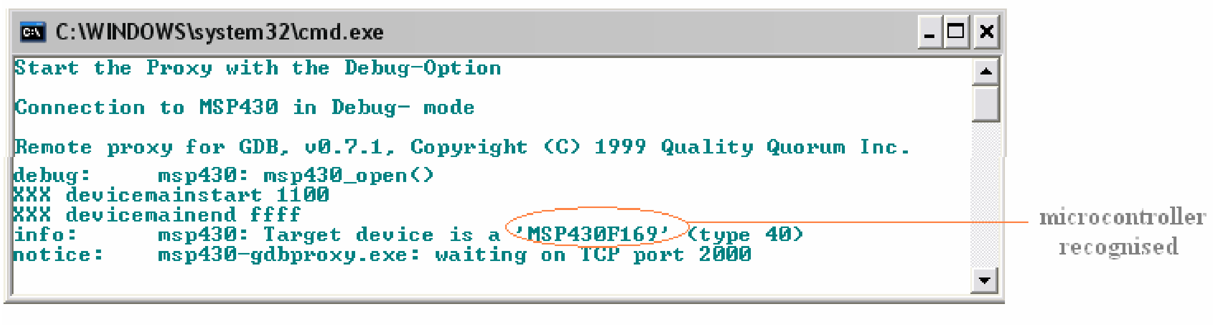
GDB stands for the GNU debugger. It is a portable debugger which runs on many Unix-like systems and works for many programming languages, including C, C++, and FORTRAN. It is free software released under the GNU General Public License.
GDB offers extensive facilities for tracing and altering the execution of computer programs. The user can monitor and modify the values of programs' internal variables, and even call functions independently of the program's normal behaviour. [24]
GDB, the GNU Project debugger, allows you to see what is going on `inside' another program while it executes -- or what another program was doing at the moment it crashed.
GDB can carry out four main functions (plus other things in support of these) to assist catch bugs in the act:
- Start the program, specifying anything that might affect its behaviour.
- Make the program stop on specified conditions.
- Examine what has happened, when your program has stopped.
- Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another. [3]

- GDB-proxy.

The program provides a TCP/IP too JTAG interface for the MSP430 FET JTAG device. The GNU remote debug protocol is respected by the TCP/IP side. As such, it could be used with GDB to provide a full in-circuit debug environment for the MSP430. The parallel port interface is supported as well as the universal serial bus box (on windows only). The proxy is started using the piece of command below.

```
@echo off
echo Start  Proxy with the  Debug-Option
echo.
echo.
echo Linking position of MSP430 in  Debug-Mode
msp430-gdbproxy.exe --debug msp430
```

As seen in Figure 5.1.1.1, the MSP430F169 has been recognised and connected. At this stage, the Debug will be executed. Should the name ‘MSP430F169’ not appear as shown in the Figure above, then hardware and or software reset will be necessary.



```
Start the Proxy with the Debug-Option

Connection to MSP430 in Debug- mode

Remote proxy for GDB, v0.7.1, Copyright (C) 1999 Quality Quorum Inc.
debug:      msp430: msp430_open()
XXX devicemainstart 1100
XXX devicemainend ffff
info:       msp430: Target device is a 'MSP430F169' (type 40)
notice:     msp430-gdbproxy.exe: waiting on TCP port 2000
```

Figure 5.1.1.1: Software connection of MSP430F169

- Giveio

The giveio driver allows programs free access to a computer’s parallel port. This driver is required when using gdbproxy on a Windows NT, 2000 or XP machine. In the absence of giveio, gdbproxy will be blocked from controlling the parallel port and consequently the JTAG tool.

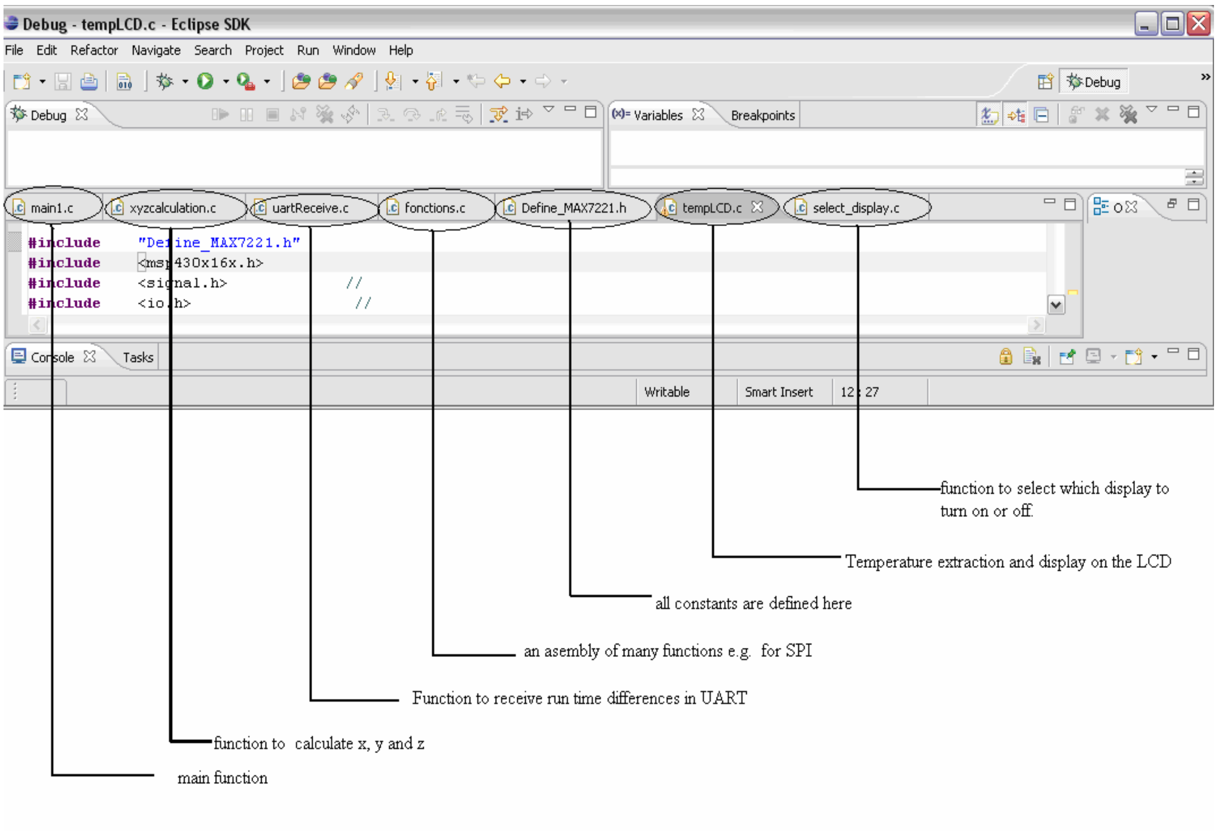
- Binutils

The GNU assembler, linker and binary utilities. The programs in this package are used to assemble, link and manipulate binary and object files. The GNU Binary Utilities, or binutils, is a collection of programming tools for the manipulation of object code in various object file formats. These include an assembler, linker and other tools for handling executable files. The current versions were originally written by programmers at Cygnus Solutions using the Binary File Descriptor library (libbfd). They are typically used in conjunction with GNU Compiler Collection, make, and GDB. [26]

5.2 –General modular software structure

In order to program deep and still have an overview of the design, a modular form is not avoidable. As seen in Figure 5.2.1, the source codes have been regrouped in modules. This structure has the advantage that it is easy for further trends, changes are easily undertaken as point of change is easily traceable and the layout is simply formal for such an amount of code.

Some of the source code designs will be analysed here, though some have already been explained in the theory section, for example the inverse calculation of a 4x4 matrix. It is easily understandable in conjunction with the source code accompanying this documentation.



```
#include      "Define_MAX7221.h"
#include      <msp430x16x.h>
#include      <signal.h>          //
#include      <io.h>               //
```

Debug - tempLCD.c - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
Debug
Variables
Breakpoints
main1.c
xyzcalculation.c
uartReceive.c
fonctions.c
Define_MAX7221.h
tempLCD.c
select_display.c
Console
Tasks
Writable
Smart Insert
12 : 27
function to select which display to turn on or off
Temperature extraction and display on the LCD
all constants are defined here
an asembly of many functions e.g. for SPI
Function to receive run time differences in UART
function to calculate x, y and z
main function

Figure 5.2.1: A snap shot illustrating Software structuring

In the main function, all the other functions are called and executed at the required time and place. The coordinates of the object are obtained in a sub function which then makes these values (x_p y_p and z_p) available in the main function. The calculation of these coordinates requires the temperature at that moment. The temperature is also calculated every cycle and as well displayed on the LCD display.
Before starting the measures, the dimension of the area of experiment is given in using the three buttons B1, B2 and B3. This is outlined in section 5.2.1.
The details of most of the important functions are described in the next sub chapters.

5.2.1 - Programming the buttons B1, B2 and B3 of the MSP430 for terminal input of length and width

There are three buttons (B1, B2 and B3) on the MSP430F169 situated on port 1 pins 5, 6 and 7 respectively. These buttons are utilised in the input of the length and width of the surface covered for determining the object found within this area. The buttons appear on the MSP430F169 as shown in Figure5.2.1.1 below. These buttons could be programmed in two manners. The first method is by using interrupt since port 1 of MSP430F169 supports interrupt. This method is generally preferable but due to certain anomalous behaviour of the interrupt, this method was considered in a second position. The second method is simply writing a function with a counter and a delay function. This function is called in the main function when the button B1 is pressed, else it is not executed. The counting is done and at the same time the value displayed on the LCD. Since only characters are sent to the LCD, a buffer of characters is saved and with the help of a ***for*** loop, these values are sent to the LCD parallel to the increment of the content of the counter. As such, the value displayed on the LCD is exactly the content of the incremented counter. The counter increments by 1, approximately after every 300ms, giving time for the user to be able to react when desired value is attained. This waiting time is both for the increment of the counter as well as for the display on the LCD. A detail description of the sending process is outlined in the section dealing with the LCD.

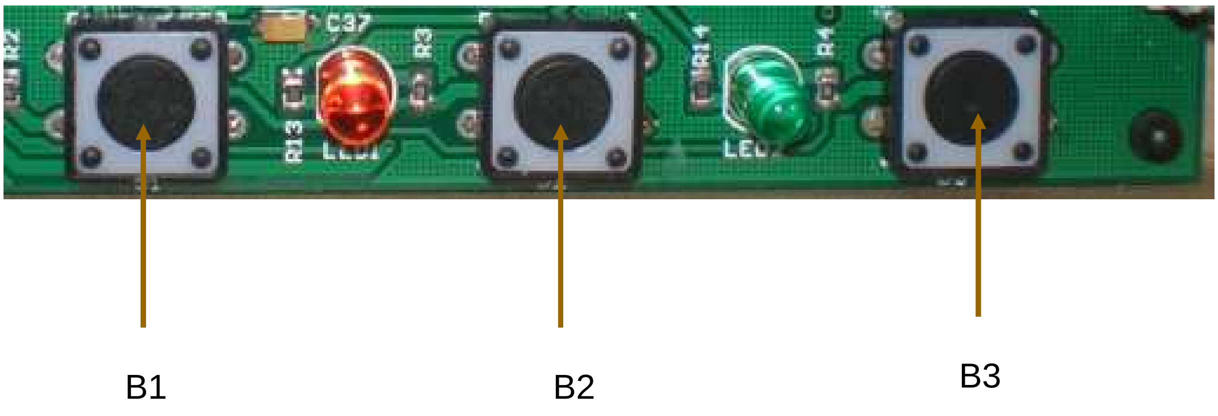


Figure 5.2.1.1: Cross section of MSP430F169 showing the three buttons

The counting is such that when B1 is pressed and not released, a function is called which functions as described above. When this button, B1 is released, a message is sent to the LCD

Figure 5.2.1.1.1 : LCD information "Width OK"

The relationship between this value read in and the surface covered by the four sensors is illustrated in Figure 5.2.1.2. Should B2 be pressed at this stage, as required, a similar procedure as for B1 starts with the difference that the content of the new counter is equals the length in metres. When the desired value is attained as seen at the LCD, the button B2 is released and a message appears indicating that the input of the length is done.
A second message appears asking the user to press B3 to exit input. When this is done, the input values are given further for calculations and the LCD set free for other purposes like displaying the temperature of the environment in relation to the temperature of a diode of MSP430F169, whose voltage fall is dependent on temperature.

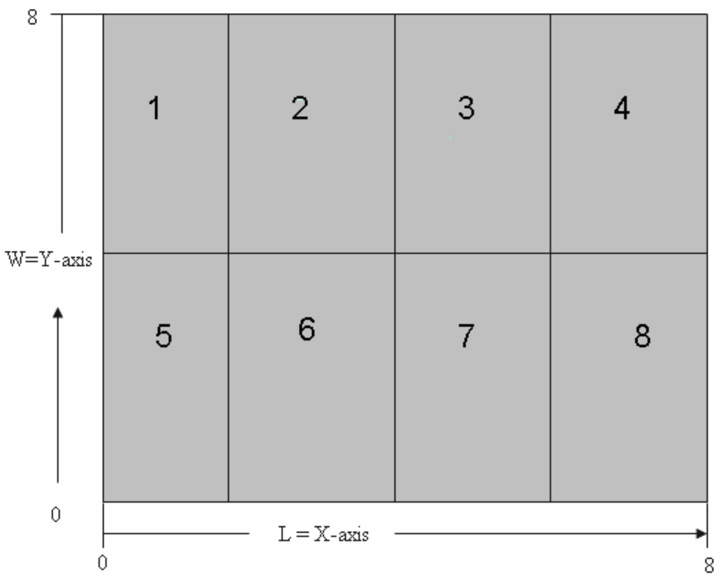


Figure 5.2.1.2: Surface area covered by sensors mapped on to the 8 LED displays

Should B2 be released when the right value is attained, a new message is displayed on the LCD indicating the input is accepted, and asking the user to press B3 to exit input process. This message keeps on blinking as long as the user does not press B3.
Button B1 and B1 with B3 for confirmation are used for the input of the width and length of the surface covered by the four sensors, which corresponds to the Y-axis and mapped to the size of two displays (See Figure 5.2.1.3) vertically to each other. These two displays could be any of the following pairs: Display 1 and 5 or Display 2 and 6 or Display 4 and 8. Button B2 is mainly used in the input of the length of the surface area, which is mapped to the length of 4 LED displays in two rows and corresponds to the X-axis. The two rows are 1-2-3-4 and 5-6-7-8. Figure 5.2.1.33 illustrates the above description.

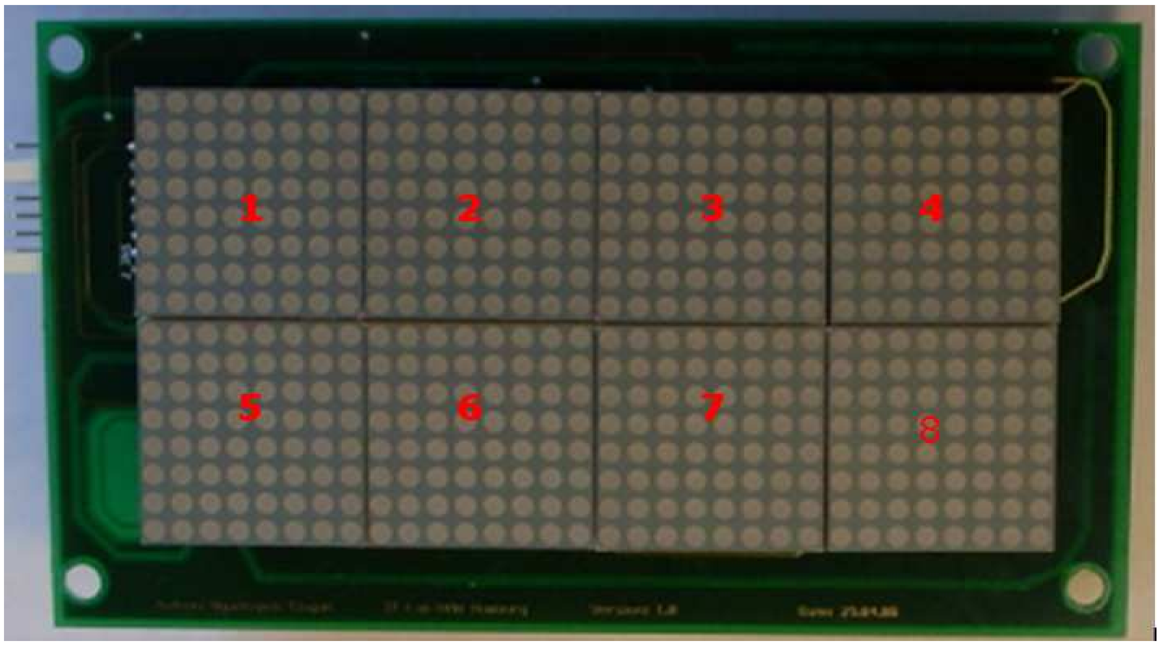
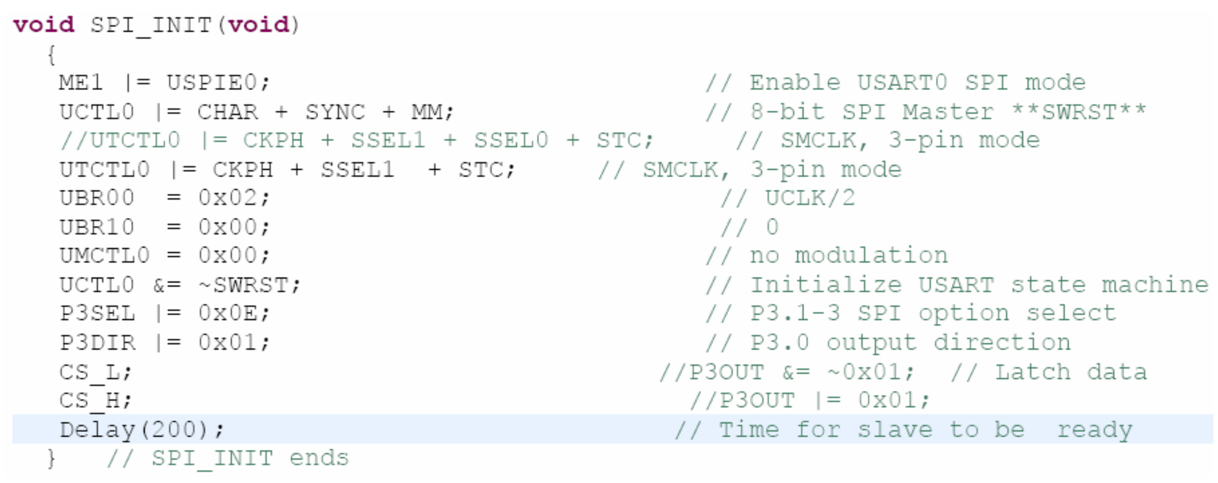


Figure 5.2.3: LED display board indicating the positions of each display

5.2.2 –LED control through USART as SPI

The ideal LED driving circuit is optimized when the characteristics of the LED are taken into consideration. LEDs are electronic devices that emit light by the delivery of current, so appropriate power sources are required to drive them. The supply voltage is typically predetermined in each application system. In this case, it is 5V. A constant-current source circuit is required to enable safe and stable lighting, designed to keep the delivery of current to LEDs at a constant rate.



```
void SPI_INIT(void)
  {
   ME1 |= USPIE0;                                  // Enable USART0 SPI mode
   UCTL0 |= CHAR + SYNC + MM;                      // 8-bit SPI Master **SWRST**
   //UTCTL0 |= CKPH + SSEL1 + SSEL0 + STC;           // SMCLK, 3-pin mode
   UTCTL0 |= CKPH + SSEL1  + STC;      // SMCLK, 3-pin mode
   UBR00  = 0x02;                                    // UCLK/2
   UBR10  = 0x00;                                    // 0
   UMCTL0 = 0x00;                                  // no modulation
   UCTL0 &= ~SWRST;                                // Initialize USART state machine
   P3SEL |= 0x0E;                                  // P3.1-3 SPI option select
   P3DIR |= 0x01;                                  // P3.0 output direction
   CS_L;                                       //P3OUT &= ~0x01;  // Latch data
   CS_H;                                         //P3OUT |= 0x01;
   Delay(200);                                   // Time for slave to be  ready
  }   // SPI_INIT ends
```

Figure 5.2.2.1: SPI Initialization

The SPI is initialised as shown in the snap shot above (Figure 5.2.2.1). There are 8 bits, no parity and one stop bit. After setting and initialising the registers, it is necessary to delay a bite to allow enough time for the slaves (MAX7221) to get ready. See Figure 5.2.2.1

5.2.3 – Software for USART as UART

The functionality of the USART as UART has already been outlined in chapter four.
The maximum correlation positions representing the run time differences for the 4 sensors are obtained through USART as UART with the help of an interrupt that automatically starts when a flag signals the begin of the send process.
The URXIFGx interrupt flag is set each time a character is received and loaded into UxRXBUF. An interrupt request is generated if URXIEx and GIE are also set. URXIFGx and URXIEx are reset by a system reset PUC signal or when SWRST = 1. URXIFGx is automatically reset if the pending interrupt is served (when URXSE = 0) or when UxRXBUF is read. The operation is shown in Figure 5.2.2.3.   [9].

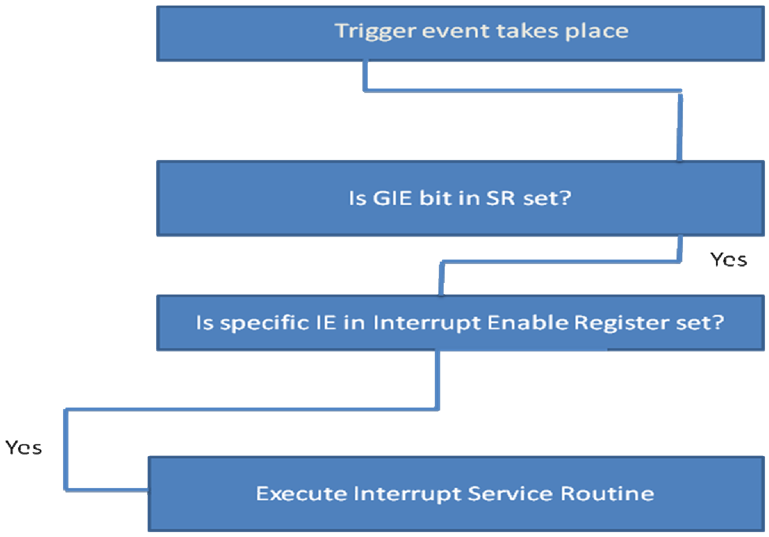

Trigger event takes place
Is GIE bit in SR set?
Yes
Is specific IE in Interrupt Enable Register set?
Yes
Execute Interrupt Service Routine

Figure 5.2.2.2: Block diagram of a typical Interrupt process

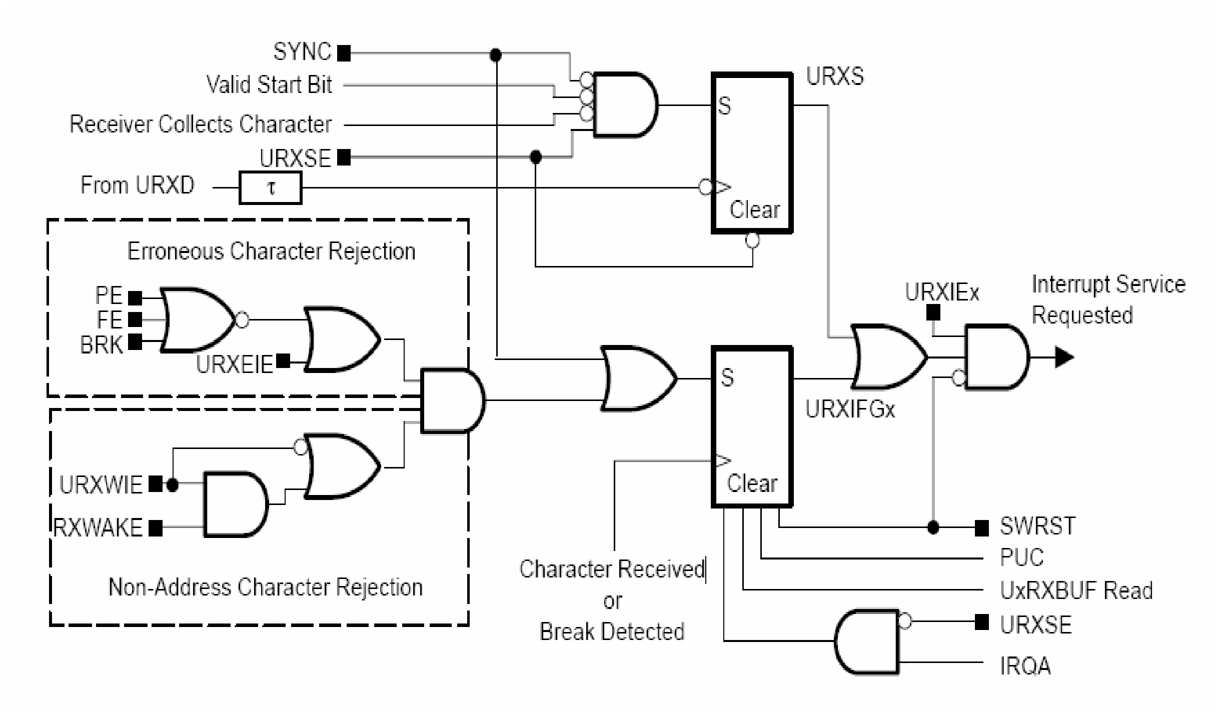


Figure 5.2.2.3: Receive Interrupt operation [9]

URXEIE is used to enable or disable erroneous characters from setting URXIFGx. When using multiprocessor addressing modes, URXWIE is used to auto-detect valid address characters and reject unwanted data characters. Two types of characters do not set URXIFGx:
- Erroneous characters when URXEIE = 0
- Non-address characters when URXWIE = 1

When URXEIE = 1 a break condition will set the BRK bit and the URXIFGx flag. [9]

5.2.4 – Software for turning on/off an LED on the display

The light emitting diodes (LEDs) are controlled with the help of the IC MAX7221 and the microcontroller MSP430. The first step in controlling the LEDs after initializing the USART as SPI of the microcontroller is represented in the flow chart shown in Figure 5.2.2.4 below.

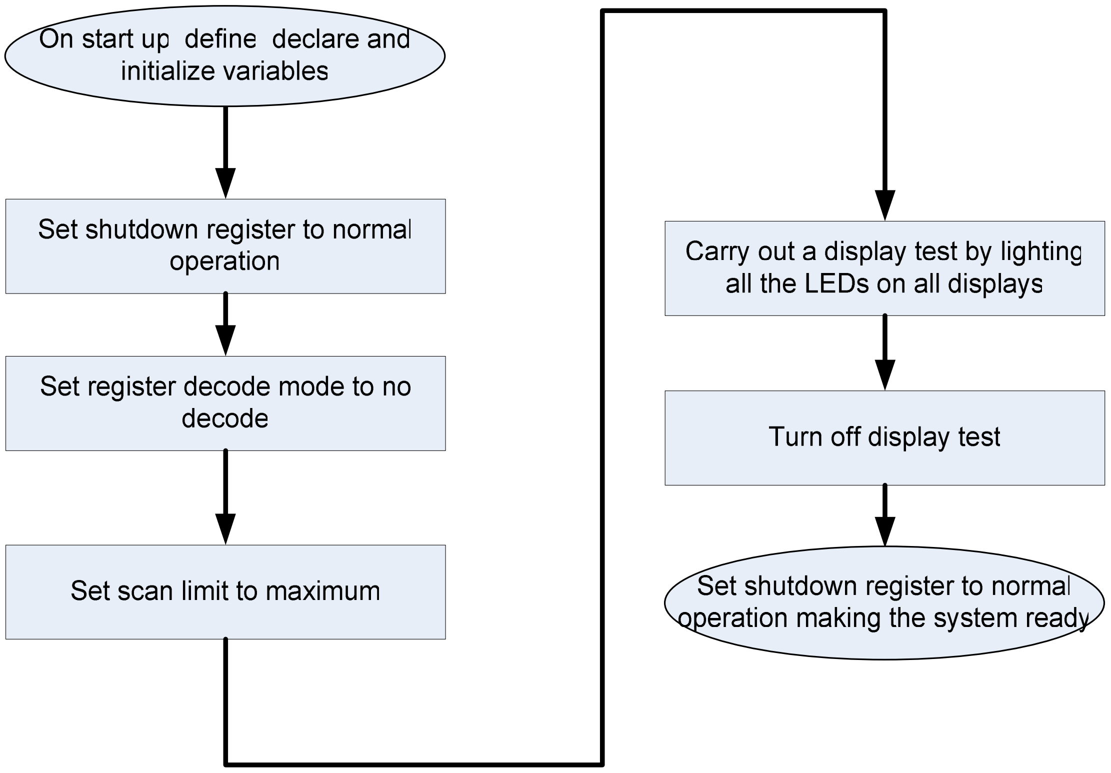
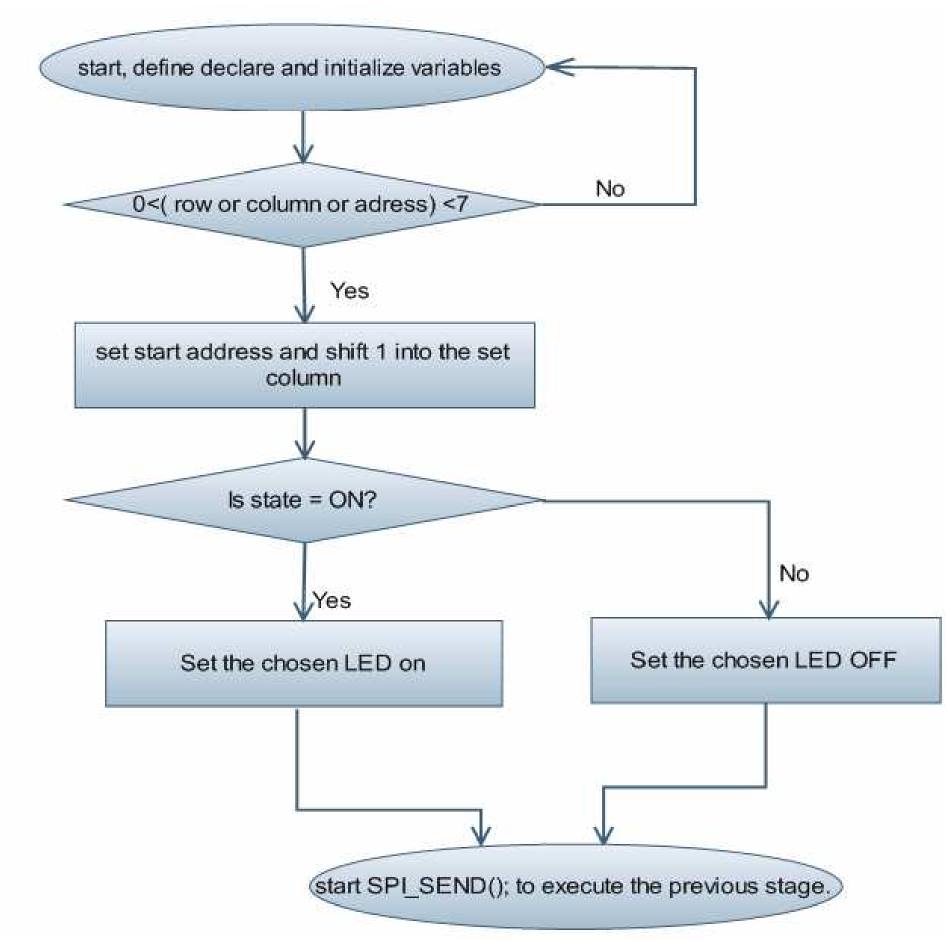

On start up define declare and initialize variables
Set shutdown register to normal operation
Set register decode mode to no decode
Set scan limit to maximum
Carry out a display test by lighting all the LEDs on all displays
Turn off display test
Set shutdown register to normal operation making the system ready

Figure 5.2.2.4: Flow chart representing the initialization of the LEDs and MAX7221

After initializing the LEDs using the MAX7221, a specific LED could be turned on, representing the coordinate of the object. This is done repetitively and fast, permitting the displacement of the object to be timely displayed. With the help of the sensitivity of the display (discussed in section 5.2.4) and modulo 8, the values to be displayed are obtained from the obtained coordinates (xp and yp). This is then displayed with the yp being the rows and the xp being the column on the display. Sensitivity helps to get a factor of multiplication on the field to represent an LED on the display. The calculation with modulo 8, helps to determine on which display the LED, as a point should be turned on or off.



start, define declare and initialize variables
0<(row or column or adress) <7
No
Yes
set start address and shift 1 into the set column
Is state = ON?
No
Yes
Set the chosen LED on
Set the chosen LED OFF
start SPI_SEND(); to execute the previous stage.

Figure 5.2.2.5: Flow chart representing the turning on /off of an LED.

It is taken care of, that the dimension of the LED (512 points) is maximally exploited with the help of sensitivity, which further assists mapping the distance on the field covered by four sensors to the number of points available on the rows and columns of the matrix display board display. The function to set an LED is one requiring three input parameters which are the address (display from 0 to 7), the row and column (from 0 to 7 each) as well as the state. The state could be ON (0x001) or OFF (0x00).

5.2.5 - Data reception through UART

Setting for UART protocol in this project was: 8 bits data length, no parity bit and one stop bit. The baud rate was set to 9600. Modulation is set and its calculation is explained with a worked example subsequently in this chapter.
When UTXEx is set, the UART transmitter is enabled. Transmission is initiated by writing data to UxTXBUF (transmission buffer). The data is then moved to the transmit shift register on the next BIT CLK after the TX shift register is empty, and transmission begins. Figure 5.2.3.1 gives more details.
The receive enable bit, URXEx, enables or disables data reception on URXDx
as shown in Figure 5.2.3.1. Disabling the USART receiver stops the receive operation following completion of any character currently being received or immediately if no receive operation is active. The receive-data buffer, UxRXBUF, contains the character moved from the RX shift register after the character is received.[9]

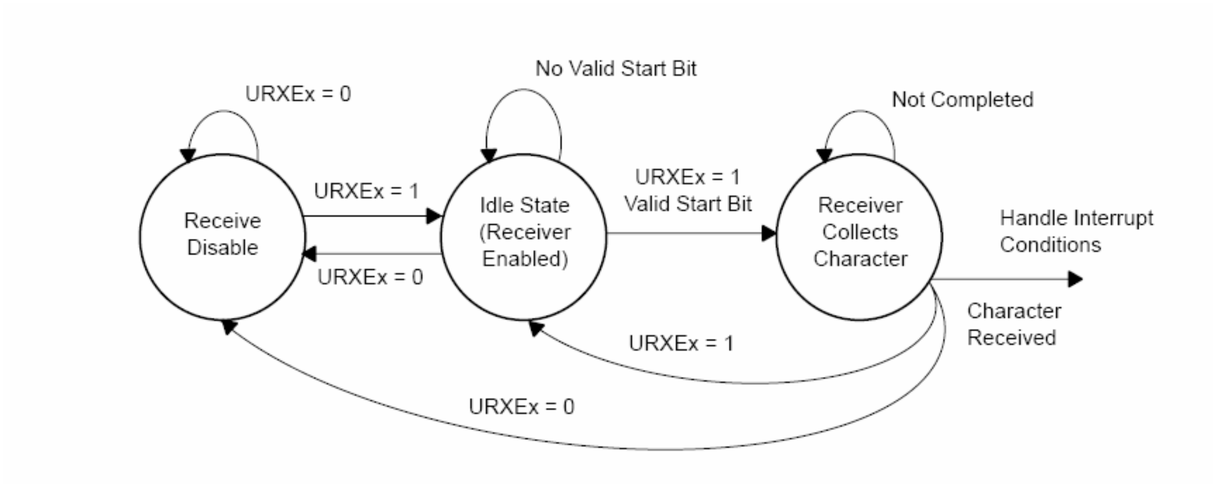
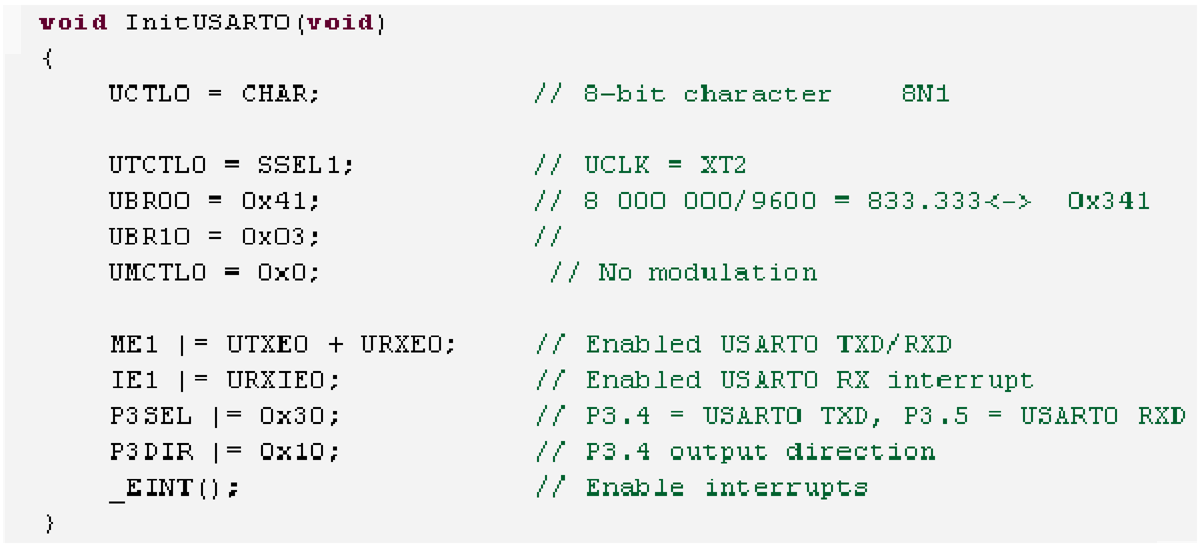


Figure 5.2.3.1: State diagram of receiver Enable [9]

When the UTXEx bit is reset the transmitter is stopped. Any data moved to UxTXBUF and any active transmission of data currently in the transmit shift register prior to clearing UTXEx will continue until all data transmission is completed. [9]



```
void InitUSART0(void)
{
    UCTL0 = CHAR;               // 8-bit character    8N1

    UTCTL0 = SSEL1;             // UCLK = XT2
    UBR00 = 0x41;               // 8 000 000/9600 = 833.333<->  0x341
    UBR10 = 0x03;               //
    UMCTL0 = 0x0;                // No modulation

    ME1 |= UTXE0 + URXE0;       // Enabled USART0 TXD/RXD
    IE1 |= URXIE0;              // Enabled USART0 RX interrupt
    P3SEL |= 0x30;              // P3.4 = USART0 TXD, P3.5 = USART0 RXD
    P3DIR |= 0x10;              // P3.4 output direction
    _EINT();                    // Enable interrupts
}
```

Figure 5.2.3.2: UART initialization

The initialization of the UART is carried out as shown in Figure 5.2.3.2. The length of the bits is 8, with no parity and just one stop bit. The clock source is then selected and the division factor set. No modulation is used in this situation as it is not necessary. The UART module is then enabled, making sending and receiving possible. The interrupt is then enabled at the end.

5.2.5.1 - Reception and data (run time differences) processing

The run time differences are the time differences between the receivers and sender. It is obtained through correlation in a parallel project and sent through universal asynchronous receiver and transmitter protocol. The settings for this protocol were 8 bit character, no parity, 1 stop bit and 9600 bits per second. These same settings are done for the sender of the maximum correlated position. The data actually sent, are the positions of maximum

correlation and vary from 0 to 127. The baud rate is given by $\text{baud rate} = \frac{\text{BRCLK}}{N}$, where $N = UxBR0 + UxBR1$ and
$UxBR0$ and $UxBR1$ are the Baud rate registers . The values and register settings are shown in the figure below.

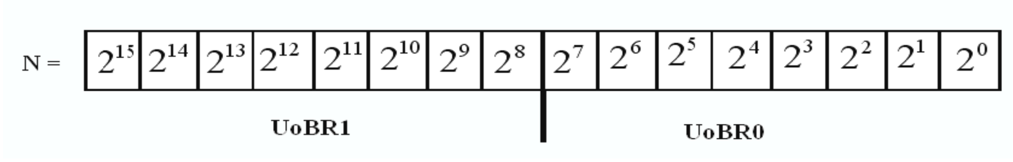


Figure: 5.2.5.1: Baud rate registers

The above mentioned positions obtained through correlation are multiplied with the period at which the signal, in this case sound was being sent. As such, the run time differences are obtained, processed and incorporated in the geometry calculation of the position of the object in question.

As discussed previously, the obtained positions through correlation are multiplied with the period of emitted signal to obtain the time differences. An example of an obtain data is shown below.
RX [17] = A034B131C160D100E
RX is the array in which the sent data which is coded is stored. There are altogether 17 characters sent for each displacement of the object. The letter A at RX [0] indicates the start of data and at the same time indicates that the maximum correlation position for the first sensor (microphones) is being sent. These are then saved in another buffer pos_t1[3]. The next character received is **0**. This is saved in pos_t1[0], the next character **3** saved in pos_t1[1] and the next character **4** is saved in pos_t1[2]. This procedure is repeated for 131, 160 and 100.The character **B** indicates the end of pos_t1[3] and the beginning of pos_t2[3], character C indicates the end pos_t2[3] and the beginning of pos_t3[3] while the character D indicates the end of pos_t3[3] and the beginning of pos_t4[3]. As already explained, pos_t$_x$[3] are buffers used to save the various positions where x corresponds to the sensor number . The character E at the last address of RX [17] signifies the end of a transmission session.
Let the obtained data **034** be considered. These values are a string of characters. The character 0 is multiplied by 100 and added to 3 which were multiplied by 10 and then the result added to 4. In order to make that clearer, the equations below are of good use.

$$pos\_t1 = 0*100 + 3*10 + 4$$

Here, i is from 1 to 3. The same procedure is carried out for pos_t2 , pos_t3 and pos_t4 to obtain the run time differences.
At this point, the data have been received and partially processed. These time differences are then added in the equation

$$\underline{\underline{p_i = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} + c*\triangle t_i}} \qquad (5.2.2)$$

$$= c*(t_i - t_0)$$

where t_i are the time differences obtained through correlation for each sensor and t_0 is the difference between the clock of the sender and that of the receiver The above equations have been programmed as explained in the theory section in chapter two.

5.2.3.1 Modulation calculation

It is worth demonstrating how the value of modulation is obtained. This aid in decreasing erroneous bits and size.

Example: a baud rate of 4800 baud is required with a crystal frequency of 32,768 Hz. This is necessary because the UART also has to run during low power mode 3. With only the ACLK available, the theoretical division factor—the truncated value is the content of baud-rate register UBR (UBR1/UBR0), given by:

$$UBR = \frac{32768}{4800} = 6.82667$$

This means that the baud-rate register UBR1 (MSBs) is loaded with zero, and the UBR0 register contains a 6. To get a rough estimate of the 8-bit modulation register UMCTL, the fractional part 0.826667 is multiplied by 8 (the number of bits in register UMCTL):

$$UMCTL = 0.82667 \times 8 = 6.613 \tag{5.2.3.1}$$

The rounded result 7 is the number of ones to be placed into the modulation register UMCTL. The corrected baud rate with the UMCTL register containing 7 ones is:

$$baud\ rate = \frac{32768}{\left(\frac{7\times 7+1\times 6}{8}\right)} = 4766.2545 \tag{5.2.3.2}$$

This results in an average baud rate error of:

$$baud\ rate\ error = \frac{4766.2545 - 4800}{4800} \times 100 = -0.703\% \tag{5.2.3.3}$$

To get the best-fitting bit sequence for modulation register UMCTL, the following algorithm can be used: the fractional part of the theoretical division factor is summed up eight times; the actual m-bit is set if a carry to the integer part occurs, and is cleared otherwise. An example using the fraction 0.82667 previously calculated follows:

Fraction Addition	**Carry to next integer**	**UMCTL Bits**
0.82667 + 0.82667 = 1.65333	Yes	m0 1
1.65333 + 0.82667 = 2.48000	Yes	m1 1
2.48000 + 0.82667 = 3.30667	Yes	m2 1
3.30667 + 0.82667 = 4.13333	Yes	m3 1
4.13333 + 0.82667 = 4.96000	No	m4 0
4.96000 + 0.82667 = 5.78667	Yes	m5 1
5.78667 + 0.82667 = 6.61333	Yes	m6 1
6.61333 + 0.82667 = 7.44000	Yes	m7 1

The result of the calculated bits m7 down to m0 is 0xEF (11101111b). The above calculations help in the amelioration of the transmission of data, as such reducing frame error.

5.2.6 – Software for displaying characters on the LCD

As already mentioned and discussed, the MSP430F169STK has an LCD display on board and can display 32 characters on two lines. This periphery has been very useful in this project. In the input of the sensor positions, the LCD display help in the confirmation of the input value. It is as well used for indicating erroneous situations. An example of such a situation is when the input value exceeds the limit. Furthermore, the LCD is used in displaying the temperature obtained from a diode of the MSP430. General information, like welcome message and others are displayed on the LCD.

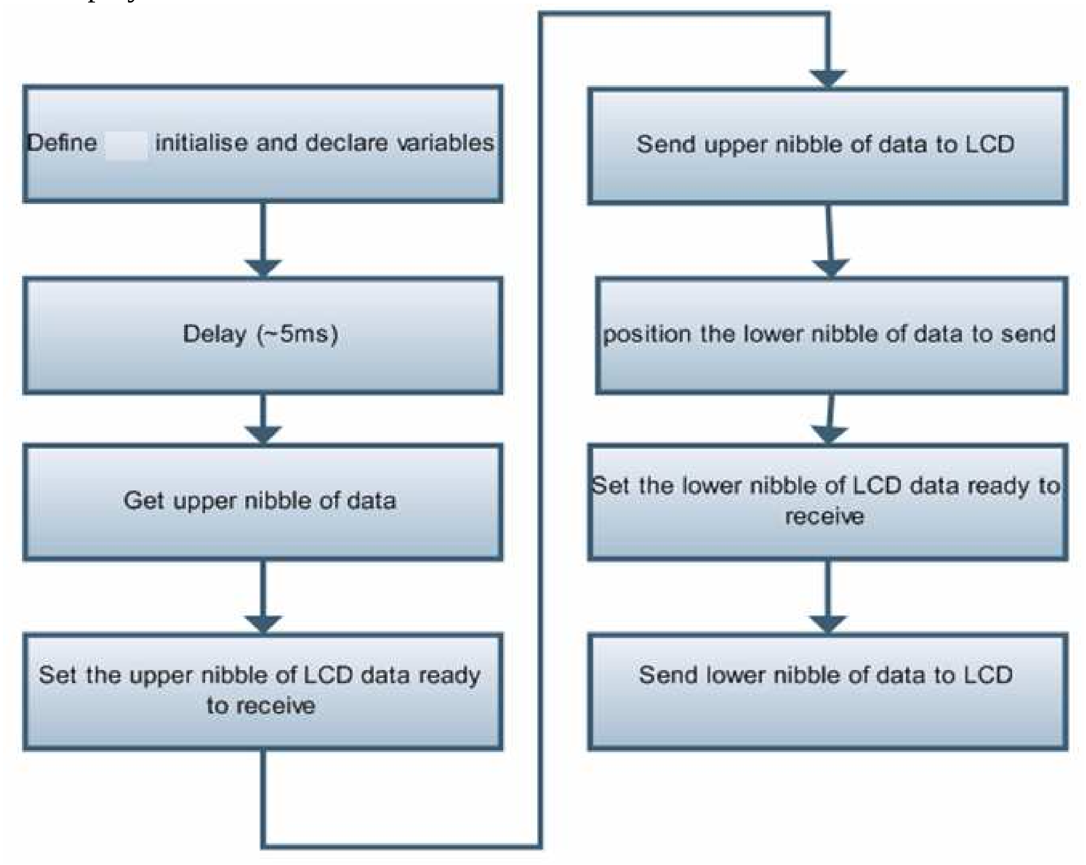

Define initialise and declare variables
Delay (~5ms)
Get upper nibble of data
Set the upper nibble of LCD data ready to receive
Send upper nibble of data to LCD
position the lower nibble of data to send
Set the lower nibble of LCD data ready to receive
Send lower nibble of data to LCD

Figure 5.2.2.5.2: Flow chart representing the displaying of characters on the LCD

The first step in writing on the LCD is to define, declare and initialise variables. It is then delayed for a while as indicated on Figure 5.2.2.5.2 so as to enable the hardware to be ready. Since sending of data is per nibble, the upper nibble of the character to be is first obtained by setting the lower nibbles to zero with an AND combination of the data with `0xf0`. The upper nibble of the LCD data is set and an OR operation is carried out with the data to be sent. This data was earlier arranged for this purpose. The same operation is carried out for the lower nibble. In addition to that, the data has to be shifted four places to the right to occupy the LSB position.

5.2.7 – Software for coordinating all functions (main function)

In the central function (`main.c`), all the other functions are called at the appropriate position, time or event. The first step is to define, declare and initialise global and local variables. The functions to initialise the LCD, UART, SPI, MAX7221 and set interrupts are called and executed. The next step is displaying a welcome message on the LCD. A for ever loop is started. It is then asked if the positions of the sensors are to be given in. This is through the three buttons as explained earlier. Should it be confirmed that the input is to be carried out by pressing B2, then a function called `buttonsinput.c` is called and the steps in executing the commands are followed. The details are outlined in the button programming section in this chapter. The above steps are represented in a flow chart in Figure 5.2.2.5.3 below.

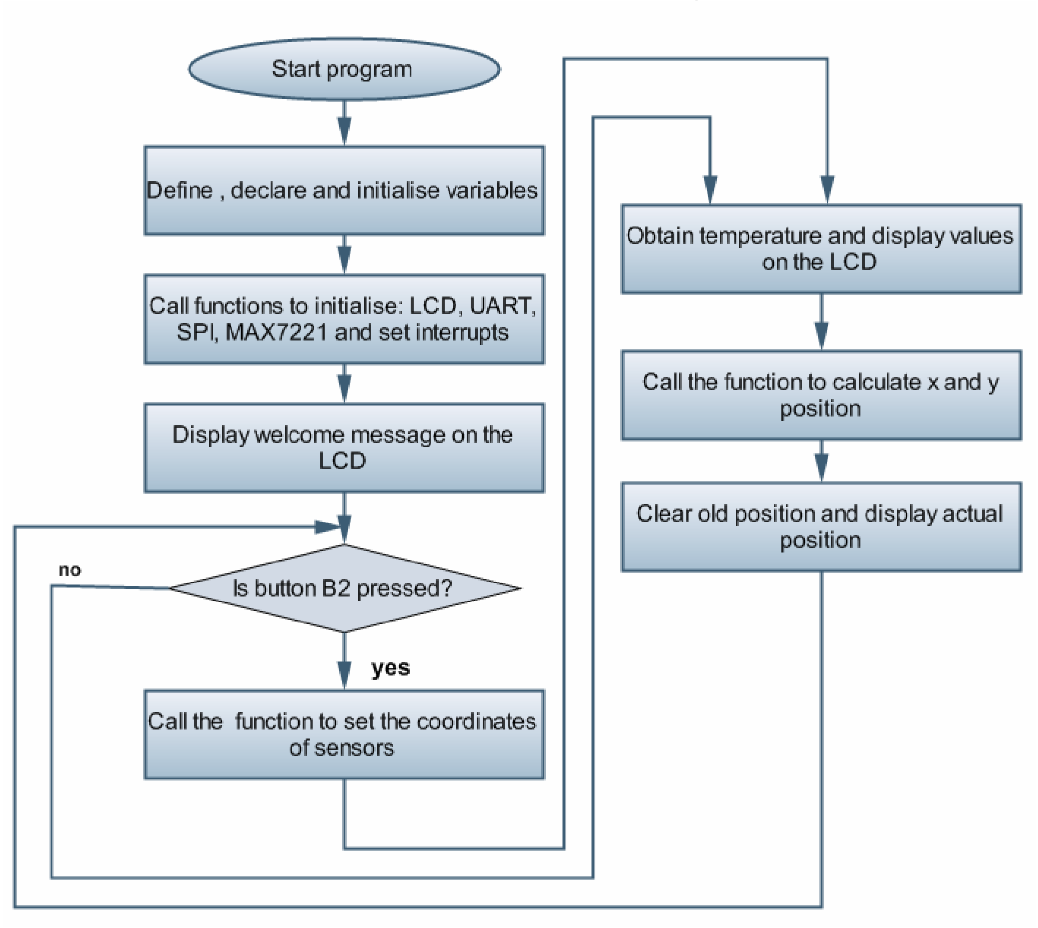

Start program
Define , declare and initialise variables
Call functions to initialise: LCD, UART, SPI, MAX7221 and set interrupts
Display welcome message on the LCD
no
Is button B2 pressed?
yes
Call the function to set the coordinates of sensors
Obtain temperature and display values on the LCD
Call the function to calculate x and y position
Clear old position and display actual position

Figure 5.2.2.5.3: Flow chart representing the main function

Should this function not be called, a next function to obtain and display the temperature is called and executed. The x and y coordinates of the object in question is then determined by calling the function called `closedform_solution();`. This function calculates the x and the y position of the object. During this process, the temperature of the environment obtained a step before is used to improve on the results as the speed of sound in air is dependent on the temperature. The final step is calling a function (`setLED(int adr, int row, int col, unsigned char state);`) that was earlier described. This function turns on an LED corresponding to the x-y pair value on the LED display. This is done as a point corresponding to the position of the object on the field. The procedures from the point where the input using B2 was done or the point where it was supposed to be done, repeats for ever and the temperature and the x and y coordinates are actualised continuously. The details of the codes are available in the CD ROM accompanying this report.

5.2.8 - Additional steps in realising the main equation in C with MSPGCC

In order to be able to realise the equation $\underline{\underline{p_i = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} + c*\Delta t_i}}$, some routines were necessary as it was difficult calling them directly and applying them. An example is the square root function (*sqr*). It was complicated calling this function directly. For this reason, it was vital programming a function that does the square root calculation. The Newton's iteration method was chosen as it gave accurate results compared with the calculator TI-92 Plus.

```
while (!(iteration++ >= 100    || x == xn)) //Make  100 iterations as long
                                                as  x!=xn

   {
     x = xn;
     xn = x - (x * x - n) / (2 * x);    // Newton's method of approximation
```

Above is a section of the code. The iteration is done 100 times as long as the result obtained is different from the result before. The result obtained is x and that before being xn. The source code is available for a detailed commentated code.

5.2.9 – Mapping of the area covered by the four sensors to the LED display

The mapping of the area under experiment to the display size is done for a few reasons. The first reason is to represent the point at which the object is located logically and easily understandable. The second point is to make use of the entire LED display board for an area between 1x1 and 8x8 square metres. The total number of LEDs to cover this range of 1x1 up to 8x8 is 512.

For a minimum length of 1metre (100 cm), a fixed distance on the field is mapped represents an LED on the display. Depending on the x- coordinate of the object, a particular LED will be lighted and the next will be lighted only when the distance per LED for that length has been covered.
This distance is obtained as such:

$$\mathrm{X_m} = \frac{L}{32} \qquad (5.2.3.1)$$

where $\mathrm{X_m}$ is the distance in the length on the field needed to be moved for a next LED on the display to be lighted and is also called the sensibility .
L is the length on the field covering the four sensors which is put in through the Button B2 on the MSP430 controller. The number 32 stands for the total number of LEDs on the four LED boards aligned next to each other on the X-axis.

For a minimum length of 1m, the distance $\mathrm{X_{m_{min}}} = \frac{1}{32} = 0.03125$ m ~ 31 mm per LED

For a maximum length of 8m, the distance $\mathrm{X_{m_{max}}} = \frac{8}{32} = 0.25$ m ~ 250 mm per LED

A similar procedure is carried out for the width as described below.

$$\mathrm{Y_m} = \frac{W}{16} \qquad (5.2.3.1)$$

where Y_m is the distance in the width on the field needed to be moved for a next LED on the display to be lighted and is also called the sensibility of the Y-axis .
W is the width on the field covering the four sensors which is put in through the Button B2 on the MSP430 controller. The number 16 stands for the total number of LEDs on the two LED boards aligned next to each other on the Y-axis.

For a minimum length of 1m, the distance $Y_{m_{min}} = \frac{1}{16}$ = 0.0625 m ~ 6 cm per LED

For a maximum length of 8 m, the distance $Y_{m_{max}} = \frac{8}{16}$ = 0.5 m ~ 50 cm per LED

The above calculated values help in covering the entire 16x32 LED matrix display for just 1m on the field or for up to 8m.

Chapter 6: Modifications to enhance quality results and some application examples

6.0 - A functional test using some fixed data

In order to carry out this test, a few positions of an object on a given surface area are chosen, the run time differences calculated and the x and y axis calculated as well. These calculated values are then compared with those obtained from the software. An example of a group of value is (`3, 2, and 1`). The digit `3` stands for the display number, which is the fourth, counting from `0`. The 2 stands for the column, where the LED to be lighted is situated or supposed to be situated. It then depends on whether it is a real or expected value. The `1` stands for the row to be lighted. In order to simplify these data, it will simply be considered as a matrix of 16 by 32. The LED point will then be given by two digits. The first stands for the x-axis and the second, for the y-axis.

$$\Delta \mathbf{t}_i = \frac{\sqrt{x^2{}_i + y^2{}_i}}{C}$$

C is the speed of sound at 25°C. This value is 346ms^{-1} . The values y and y are estimated or measured and $\Delta \mathbf{t}_i$ subsequently calculated. The conditions mentioned above are for the expected values. The temperature is not stable, though roughly 25°C. This difference in temperature values affects the real values. These could be observed in the tables below.

A test for an 8m by 8m surface

For a surface of 8m by 8m, the following results were obtained.

x in m	y in m	Δt_1 in µs	Δt_2 in µs	Δt_3 in µs	Δt_4 in µs	Expected position	Real position
1	8	2890	20231	23301	30723	(4,16)	(4,16)
2	7	6463	17580	21041	26646	(8,14)	(8,14)
3	6	10421	15564	19388	22573	(12,12)	(12,12)
4	4	16349	16349	16349	16349	(16,8)	(16,8)
5	5	16852	12262	20437	16852	(20,10)	(20,10)
6	3	22573	15564	19388	10421	(24,6)	(24,6)
7	2	26646	17580	21041	6463	(28,4)	(28,4)
8	1	30723	23121	23301	2890	(32,2)	(32,2)

Table 6.0.1: Functional test for 8m by 8m

Table 6.0.1 above shows the characteristics properties for a field of 8 meters squared. The real values are exactly the expected values. This is because it is a simulation and does not involve rounding up of digits that could result in differences between real and expected values.

A test for a 4m by 8m surface

For a surface of 4m by 8m, the following results were obtained.

x in m	y in m	Δt_1 in μs	Δt_2 in μs	Δt_3 in μs	Δt_4 in μs	Expected position	Real position
1	8	2890	8671	23301	24694	(8,16)	(8,16)
2	7	6463	6463	21041	21041	(16,14)	(16,14)
3	6	10421	6463	19388	17580	(24,12)	(24,12)
4	4	16349	11561	16349	11561	(32,8)	(32,8)
3	5	12262	9140	16852	14737	(24,10)	(24,10)
2	3	15564	15564	10421	10421	(16,6)	(16,6)
1	2	17580	19388	6463	10421	(8,4)	(8,4)
1	1	20437	22011	4087	9140	(8,2)	(8,2)

Table 6.0.2: Functional test for 4m by 8m

Table 6.0.2 shows calculated values and expected values for a surface of 4m by 8m. The values are exactly as expected. These are due to the fact that it is a simulation and does not involve fractions leading to rounding up higher decimal places to null. In such a situation, there will be a deviation of the real values from the expected values.

A test for a 4m by 6m surface

For a surface of 4m by 6m, the following results were obtained.

x in m	y in m	Δt_1 in μs	Δt_2 in μs	Δt_3 in μs	Δt_4 in μs	Expected position	Real position
1	6	2890	8671	17580	19388	(8,16)	(8,16)
2	5	6463	6463	15564	15564	(16,13)	(16,13)
3	4	10421	15564	19388	22573	**(24,11)**	**(24,10)**
4	3	10421	6463	14451	11916	(32,8)	(32,8)
1	2	11916	14451	6463	10421	(8,5)	(8,5)
2	1	15564	15564	6463	6463	**(16, 3)**	**(16,2)**

Table 6.0.3: Functional test for 4m by 6m

In Table 6.0.3 the results obtained confined to the expected results, but for the x-y positions (3, 4) and (2, 1). This is due to rounding up error which could be corrected by watching the fractions and consequently correcting the values.

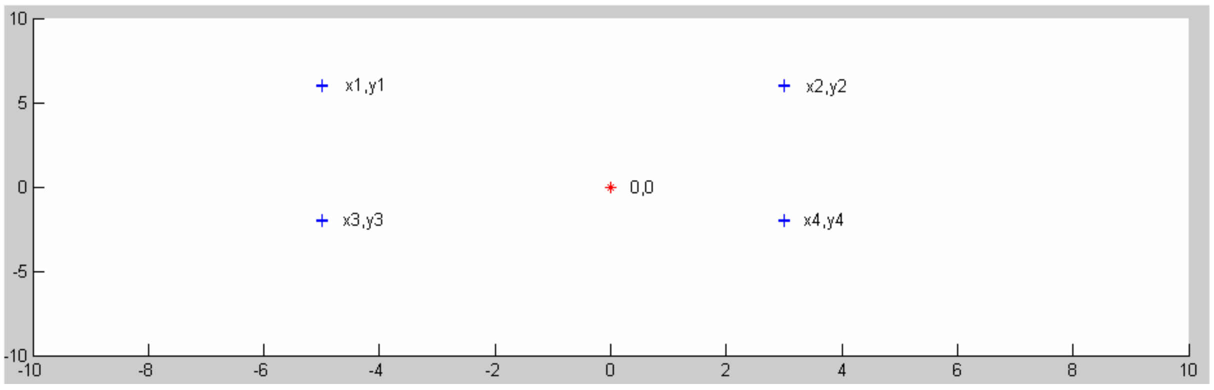


Figure 6.0.1: Simulated results in Matlab demonstrating the object and the four axes on the surface

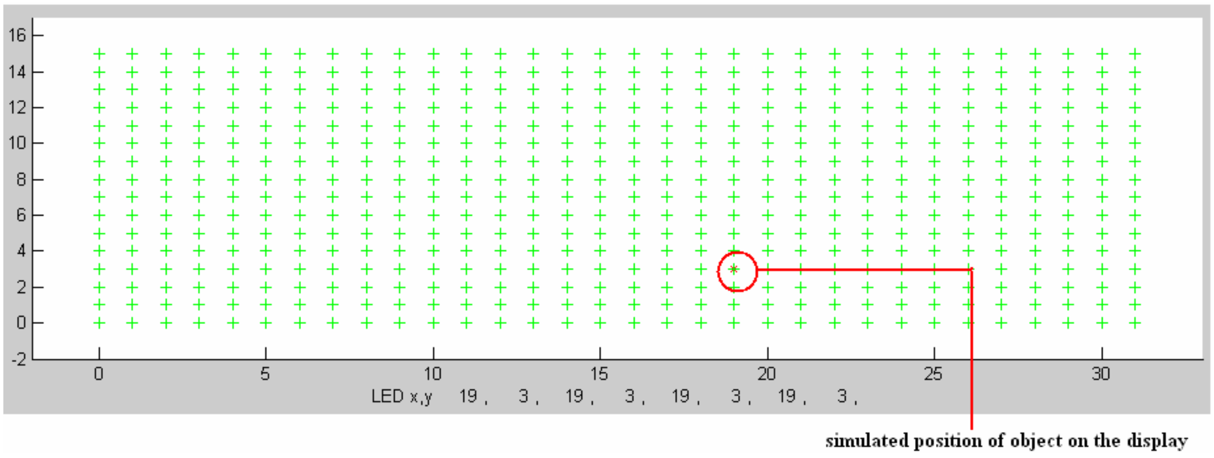


Figure 6.0.2: Simulated position in Matlab displaying (19, 3) on the matrix display

Figure 6.0.1 above shows the field of experiment, with the axes and the object in question. Figure 6.0.2 shows a simulated example of the position of an object. The method used in this simulation is the closed solution method with the assumption that the axes are in a rectangular form. This display could be compared with Figure 6.0.1. This figure is the actual position of the object as on the field. In the case where there is an error or the results are not accurate, there is more than one point lighted on the LED.

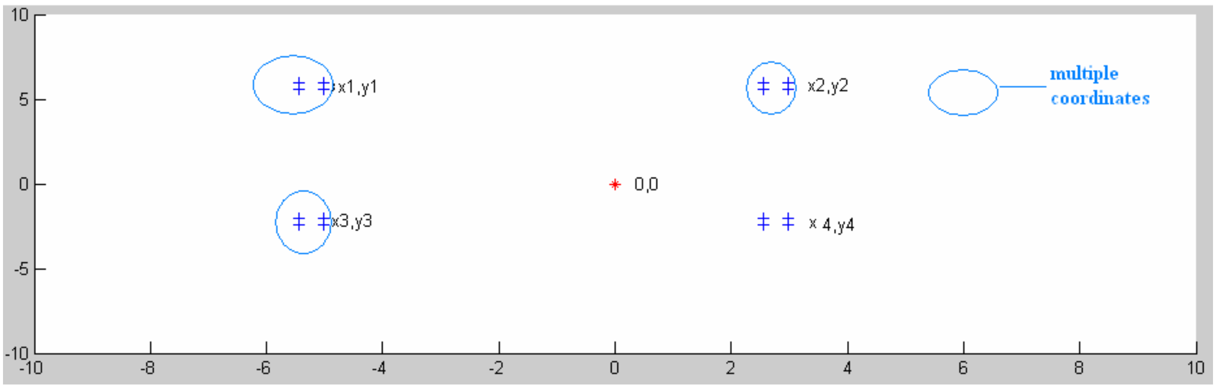


Figure 6.0.3: Erroneous simulated position on the field

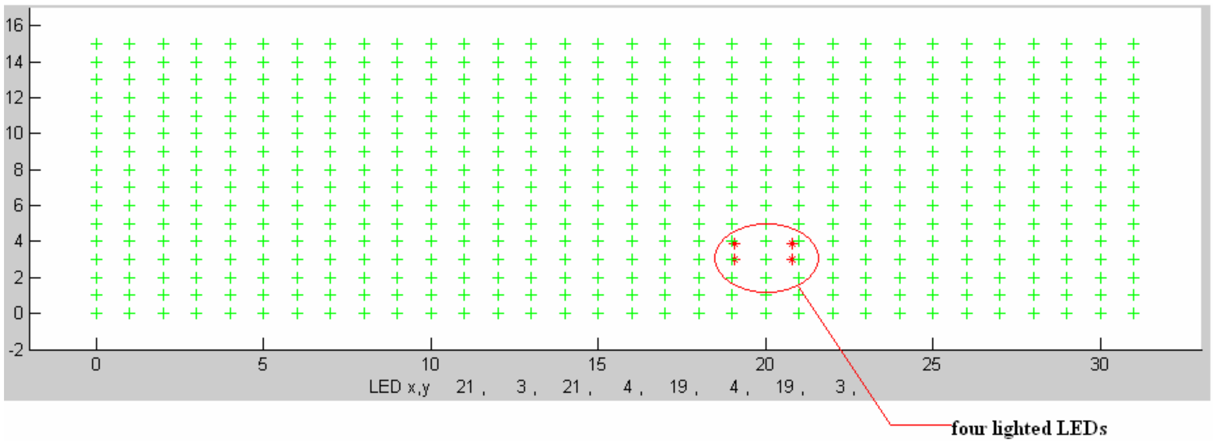


Figure 6.0.4: Erroneous simulated position on the display

Figure 6.0.3 and 6.0.4 show some erroneous situations. These situations are built in the simulation to reflect some extreme real situations where there is for example a lot of reflection of sound or other forms of disturbances. The factors influencing the system have been tackled appropriately starting from reflection (in another sub project) to temperature changes

The modifications to improve on the results, which have been outlined alongside the methods, could be divided in to two sections. The first section is the adequate management of the software to improve on the performance of the microcontroller and the second is the modification of some parameters influencing the results like the temperature of air, reflection of sound and outlier.

6.1 – Improving Speed, Performance and reducing Power Consumption of MSP430

There are four low power modes in addition to regular operating mode on the MSP430:
Active Mode is the fully powered mode when the processor executes code and all clocks and peripherals are active. The chip consumes about 340 µA with 1 MHz clock at 3.3V in this mode.

i -Low Power Mode 1 (LPM1) disables the CPU and MCLK while leaving the ACLK and SMCLK enabled. This allows timers, peripherals, and analogue systems to continue operation while dropping current consumption to about 70 µA with 1MHz clock at 3.3V. Because the timers and other internal interrupt systems still operate, the processor will be able to wake itself.

ii- Low Power Mode 2 (LPM2) disables the CPU, MCLK, and the DCO are disabled but the SMCLK and ACLK are active. The DC is disabled if the DCO is not used for MCLK or SMCLK in active mode. Internal interrupts can still operate. Current consumption drops to about 17 µA.

iii- Low Power Mode 3 (***LPM3***) disables the CPU, MCLK, SMCLK, and DCO. The DC and ACLK remain active. This allows some peripherals and internal interrupts to continue. Current consumption drops to about 2 µA.

iv – Low Power Mode 4 (***LPM4***) Current consumption drops to about .1 µA, but all clocks and the CPU are disabled. This prevents any of the on-chip modules from operating, and only off-chip interrupts can wake the device.

To enter a low power mode the status register in the CPU has been set to indicate the desired mode. This change of mode helps considerably in the reduction of power consumption. The long life of the battery is achieved through judicious hardware and software design. Skilful programming will allow the same work to be done with cheaper parts to improve the bottom line. It is well known from the consumer computer market that the speed of computers can be measured in hertz (Hz). It is less well known that the frequency of the computer's processor does not adequately indicate a computer's performance or even the performance of the processor itself. In order to improve the performance of a software application, it is necessary to understand the way performance is measured.

A simple way to get the time a program will take to perform a task is to count the number of processor cycles that the code will take. On the MSP430, each CPU instruction, jump, and interrupt takes a fixed number of cycles as explained in the MSP430 user's Guide. Taking into account branching, function calls, and interrupts, the assembly code of a program can be used to calculate the time needed for a section of code. If this time is known, alternative methods could then be used to find out which takes a shorter time to do the task. This method was adapted in the software development. This led to a faster task performance and less battery consumption. The battery consumption is an important factor as the device is a stand-alone requiring battery as voltage supply and not an AC source.

Removing unnecessary instructions is a start to improving performance. Test code left in a final version, any unnecessary instructions in a loop, can all significantly increase the time in a section of code.

In C, unnecessary code takes the form of too many method calls inside of a loop (because each method call adds a layer to the heap). While this is only a slight efficiency loss for code that is only executed once per sample, the loss can be very damaging if multiplied by a large loop. When trying to reduce execution time, it is best to start with the regions of the code where the processor spends the most time. Parts of the program that are only executed rarely have only a small effect on the speed compared to a loop that might run 100 times per sample. For example, if a duty can be done once, outside of the loop, do not do it many times inside of the loop. [14]

A judicious use of timers and other instruction saving interrupts were taken in consideration. The timer interrupts allow the processor to periodically check on the status of the program without the use of slow ***`while(1)`*** or ***`for(;;)`*** loops (polling). However, for correct program behaviour, it was important doing the minimum possible work in an interrupt. This is most important with interrupts that happen frequently because the control flow of the program can be thrown off when interrupts happen faster than the system can handle. If the same interrupt occurs a second time before the first occurrence of the interrupt has completed, program behaviour is much more difficult to control. It is much easier to simply ensure that the interrupt is short enough to avoid the danger all together.

It was also avoided to calculate values that have been calculated already. A piece of reusable information was saved and simply called later.

6.2 –Temperature and influence on the speed of sound in air

A further factor in ameliorating on the quality of the results is the incorporation of temperature to obtain an accurate value of the speed of sound in air at the moment of measurement.

A sound wave is a pressure disturbance which travels through a medium by means of particle-to-particle interaction.

The speed of sound is variable and depends mainly on the temperature and the properties of the substance through which the wave is travelling. In this project, the medium is air.

Since temperature and thus the speed of sound normally decrease with increasing altitude, sound is refracted upward, away from listeners on the ground, creating an acoustic shadow at some distance from the source. Figure 6.2.1 shows a sound source and the manner in which it vibrates in air.

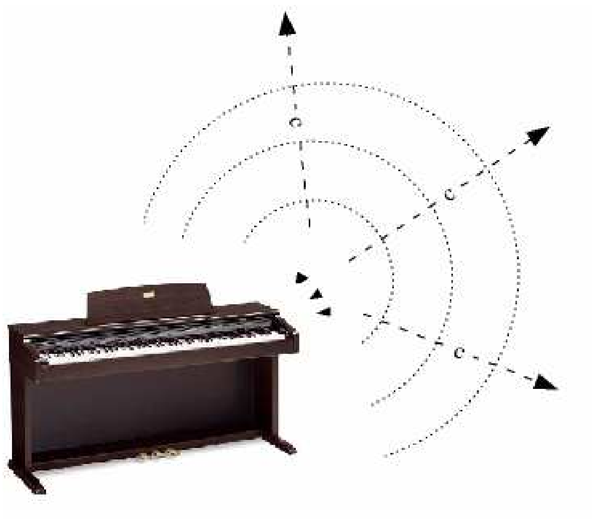


Figure 6.2.1: Sound vibration in air.

The decrease of the sound speed with height is referred to as a negative sound speed gradient. However, in the stratosphere, the speed of sound increases with height due to heating within the ozone layer, producing a positive sound-speed gradient.
All other things being equal, sound will travel more slowly in denser materials, and faster in stiffer ones. For example, sound will travel faster in iron than uranium, and faster in hydrogen than nitrogen, due to the lower density of the first material of each set. At the same time, sound will travel faster in iron than hydrogen, because the internal bonds in a solid like iron are much stronger than the gaseous bonds between hydrogen molecules. In general, solids will have a higher speed of sound than liquids, and liquids will have a higher speed of sound than gases. The variation of the speed of sound and the density (different medium) is given in the equation below.

$$c = \sqrt{\frac{K}{\rho}} \qquad (6.2.1)$$

Where K is a coefficient of stiffness ρ is the density.
In this project, it is not of great interest to analyse the behaviour of the speed of sound in different mediums since air alone is the medium here.

$$C = 331.3ms^{-1}\sqrt{1+\frac{\vartheta}{273.15}} \qquad (6.2.2)$$

From the above equation, it is further simplified to equation 6.2.2.1 below.

$$C = 331.3 + 0.606 * \vartheta ms^{-1} \qquad (6.2.3)$$

where ϑ is the temperature in degrees Celsius (°C) and C, the speed of sound at this temperature.

Speed of sound in air in m/s	**Temperature in °C**
331.30	0,0
337.360	10,0
343.42	20
346.450	25
349.480	30
355.540	40
361.60	50
367.660	60
373.720	70
379.780	80
385.840	90
391.90	100

Table 6.2.1: Table of variation of speed of sound in air with temperature.

Table 6.2.1 shows how the speed of sound in air is related to the temperature. For this demonstration, values are taken between 0 and 100 with an interval of 10, except at the point 25°C (room temperature) which is important. At this point, the speed of sound is 346.450 m/s. This variation of speed of sound with temperature in the air is worth analysing as it improves the accuracy of the coordinates of the object in question.

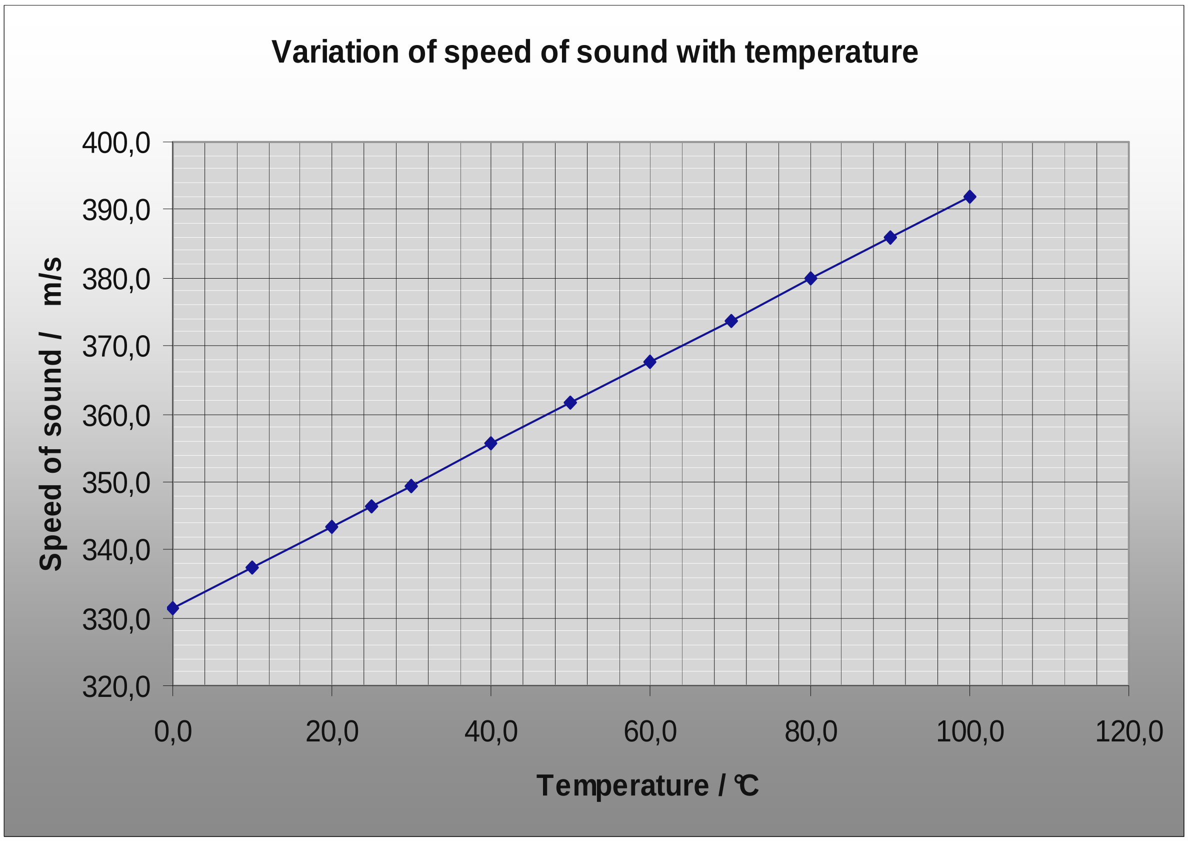


Figure: 6.2.2: Variation of speed of sound in air with temperature

The gradient of the curve gives the 0.6ms^{-1}/°C constant and the intercept being ~331ms^{-1}. This confirms the linear behaviour of the speed of sound – temperature of air. [27]

6.3 – Obtaining the temperature using MSP430F169 Starter Kit

The MSP430F169 Starter Kit makes it possible to obtain the temperature of the air at the place and time of experiment. There is a diode on board of this microcontroller whose change in voltage is directly related to the temperature of the diode. This change of voltage yielding the temperature of the diode could be calibrated to give the temperature of the room or place of carrying out the measurements. MSP430F169 offers a precision analogue-to-digital converter delivering an accuracy of about 1°C. There are tow ADC: the 10 bits resolution ADC (ADC10) and the 12 bits ADC (ADC12). This periphery is one of the most sophiscated peripheries of MSP430F169 though requiring less power in realising its task of conversion as it works with the CPU asleep. In this project, the ADC12 is employed for the mare fact that it offers better results compared to the ADC10 with less resolution. ADC12 has a built in sample-and-hold circuit. [9] The front end is made up of a multiplexer circuit. This circuit makes it possible to choose one of eight external pins. The voltage fall in relation to temperature in °C is given by 3.35mV/°C for this microcontroller. Since the voltage fall over the diode is linearly related to the temperature of the diode, it is then calibrated at about 25°C to obtain the room temperature. More details, especially concerning the various modes are explained in the ADC section (chapter 3 sub section 3.3). It is to be taken note of that the calibration varies from one MSP430F169 to the next.

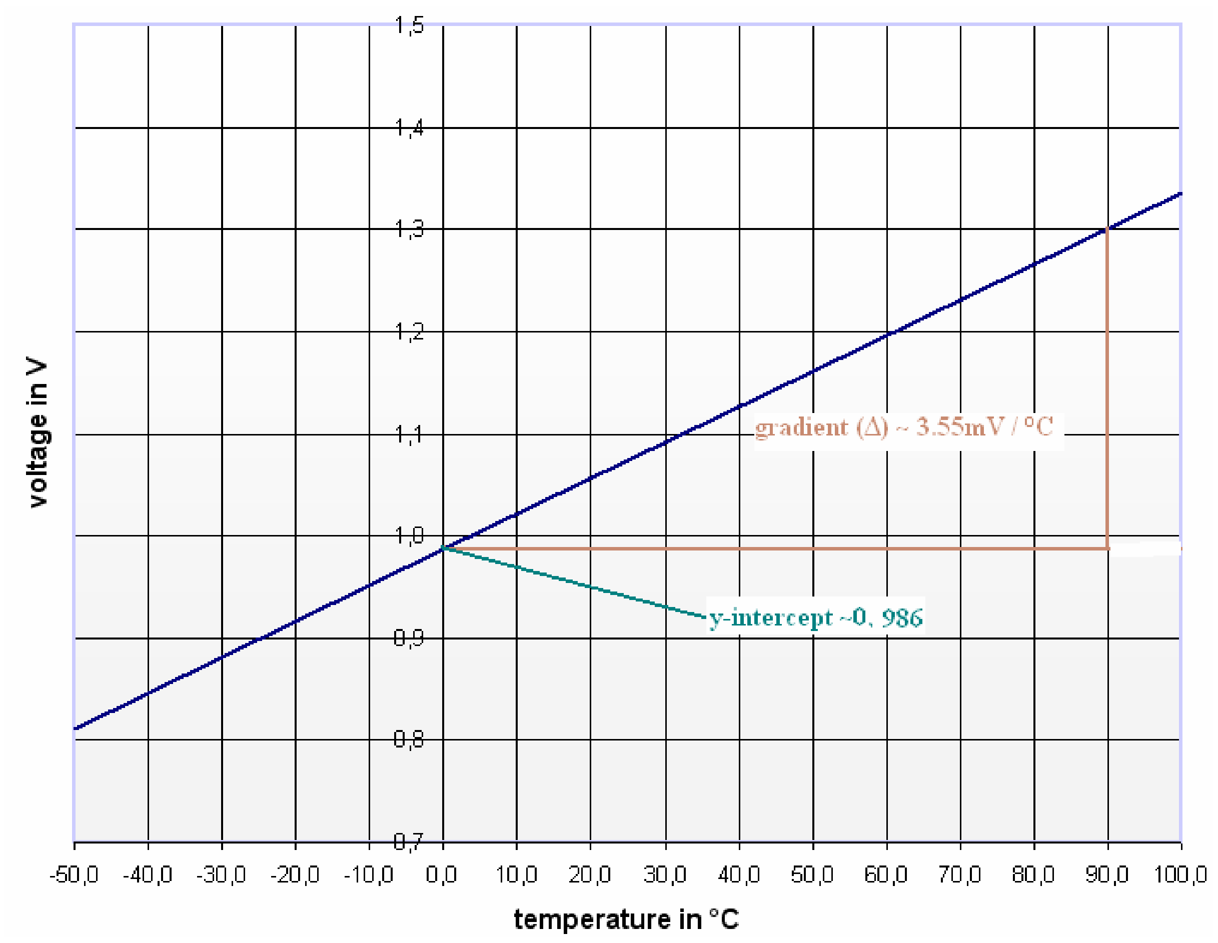


Figure 6.3.0: Typical temperature transfer function for MSP430.

A typical temperature transfer function could be represented as in Figure 6.3.0 above. It is a linear function with the y-intercept of ~ 0.986 and a positive gradient of $3.55*10^{-3}$ V/°C.

Figure 6.3.3 describes the procedures in obtaining the temperature of the environment in which the position determination is being carried out.

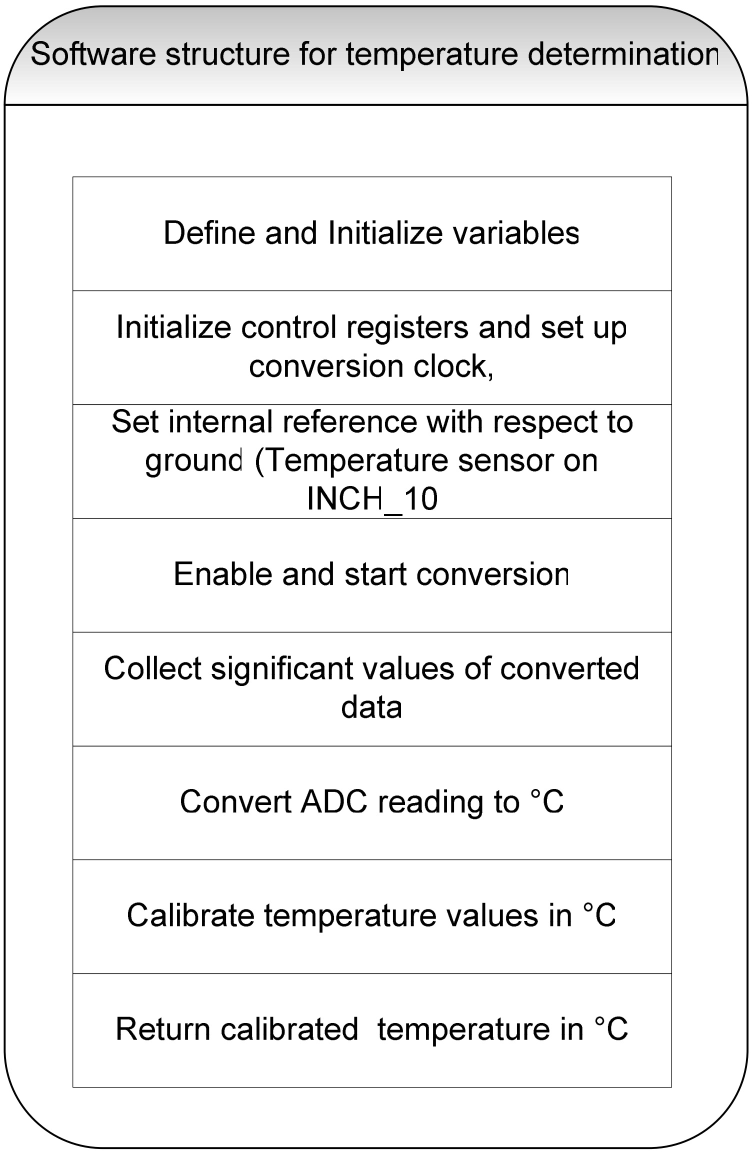

Software structure for temperature determination
Define and Initialize variables
Initialize control registers and set up conversion clock,
Set internal reference with respect to ground (Temperature sensor on INCH_10
Enable and start conversion
Collect significant values of converted data
Convert ADC reading to °C
Calibrate temperature values in °C
Return calibrated temperature in °C

Figure 6.3.1: software structure for temperature determination using MSP430F169

The elaboration of ADC in chapter three, section 3.3 gives more information for temperature determination using MSP430.
In writing the code, it is considered [9], tested and proven that there is a voltage fall of 3.35mV per °C. Figure 6.3.0 above supports this statement. This measurement gives an accuracy of about 1°C. It is enough to obtain better results for this project. Should there be a necessity for a more accurate measurement, then the idea of using an external sensor could have been analysed.
The first step is to define and initialize variables as well as the necessary registers. The voltage reference is then set. This reference is internal. As next, the ACD12 is started. As long

as the conversion is going on, the next instruction is not executed until conversion is finished. During this time of conversion, the CPU could be doing other activities. As soon as the conversion is over, the last 12 bits of the converted value being voltage fall is retrieved and saved in a temporal memory for further calculations. The reason why only the last 12 bits are considered is because it is enough as far as the accuracy is concerned and voltage fall is in that range.
The content of the temporal memory is now multiplied with a constant (standard for this MSP430 type) **a** and divided by another constant **b** by shifting it **c** times to the right.
The value now undergoes calibration to obtain the room temperature in °C.
The few lines of code below give more details.

```
temperatu= ADC12MEM0 & 0x00000FFF;
temperatu  *= 845;  // a=845  Steps to convert to degree
temperatu >>=13;  // => divided by 8192= b  Calibration
purpose
                       //temperatu = 278;  13= c
temperatu -= (278+6);  // 6 is due to calibration
```

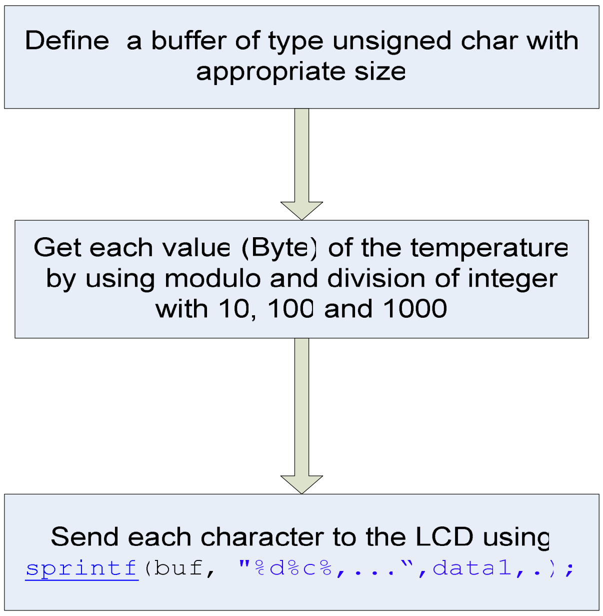

Define a buffer of type unsigned char with appropriate size
Get each value (Byte) of the temperature by using modulo and division of integer with 10, 100 and 1000
Send each character to the LCD using
sprintf(buf, "%d%c%,...",data1,.);

Figure 6.3.2: Block diagram showing steps to prepare data to display

The obtained temperature is broken down into single Bytes and sent to the LCD display as seen in Figure 6.3.2. The first step is to define an array (buffer) that can contain all the characters to be displayed. These values are then saved at various addresses of the buffer and as such with a loop, they are sent singly to the LCD. This procedure is repetitive in the main function, delivering the actual temperature every few milliseconds.

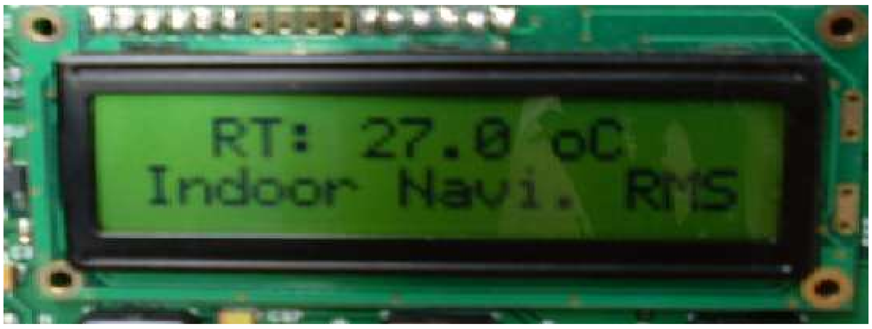


Figure 6.3.3: Room temperature (hot summer day) in °C displayed on the LCD

The obtained temperature is displayed on the LCD. See Figure 6.3.3 above. R,T, **:**, space,2,7,.,0,space,o,C, and the rest of the data are sent singly to the LCD . A single character is sent by sending two times a nibble. The upper nibble is first sent, followed by the lower nibble. As such it is possible to make almost any character appear on the LCD.

6.4 - Using the Oscilloscope to control or verify SPI Data.

In the primordial phase of SPI programming, the sent Data were verified on the Oscilloscope so as to adjust the settings/ initialization of the SPI-Periphery. The CLK-Signal and the Data-Signal from the MSP430F169 were connected to Channel 1 and 2 of the oscilloscope (Tektronix TDS744A). After a setup as seen on the figure below, the results could be obtained. The data sent was 0x55 (in binary form; 01010101) and the clock was set at 4MHz. After obtaining the results, the configuration of the periphery was maintained and its programming continued.

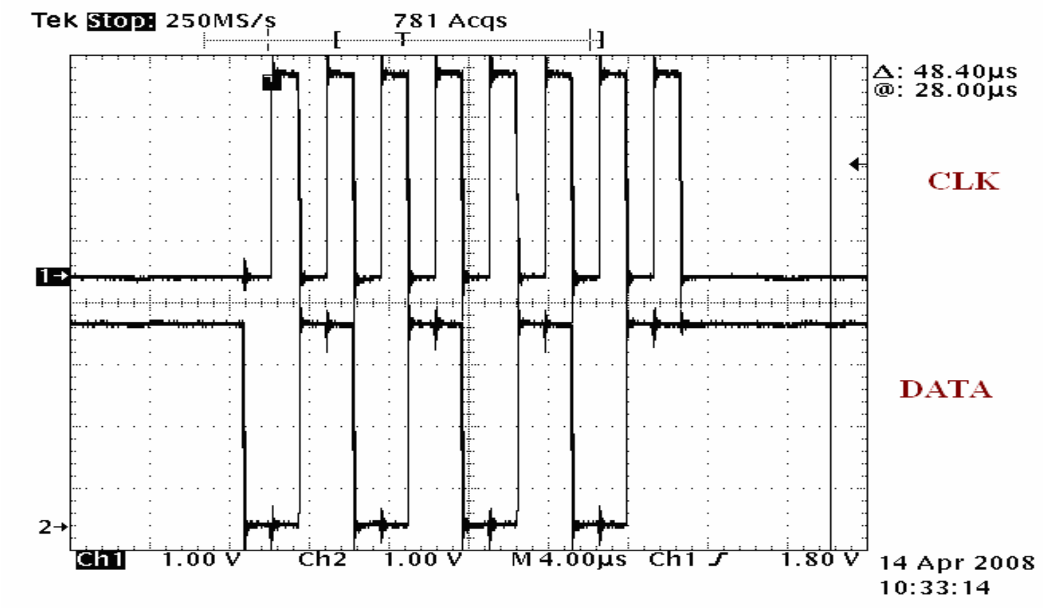


Figure 6.3.4: SPI results from Oscilloscope Tektronix TDS744A

The principles are such that Data bits are inverted, that is +V means 1/H (H: High) and –V means 0/L (L: Low). The Bits in a Byte are transmitted such that LSB is first and MSB last. It is also called “little endian”.

Further more, each byte begins with a high start bit and ends with one or two stop bits that are low. In this case, there is one stop bit, 8 bits data length and no parity bit. The line is low when idle. Figure 6.3.5 shows the above explanation.

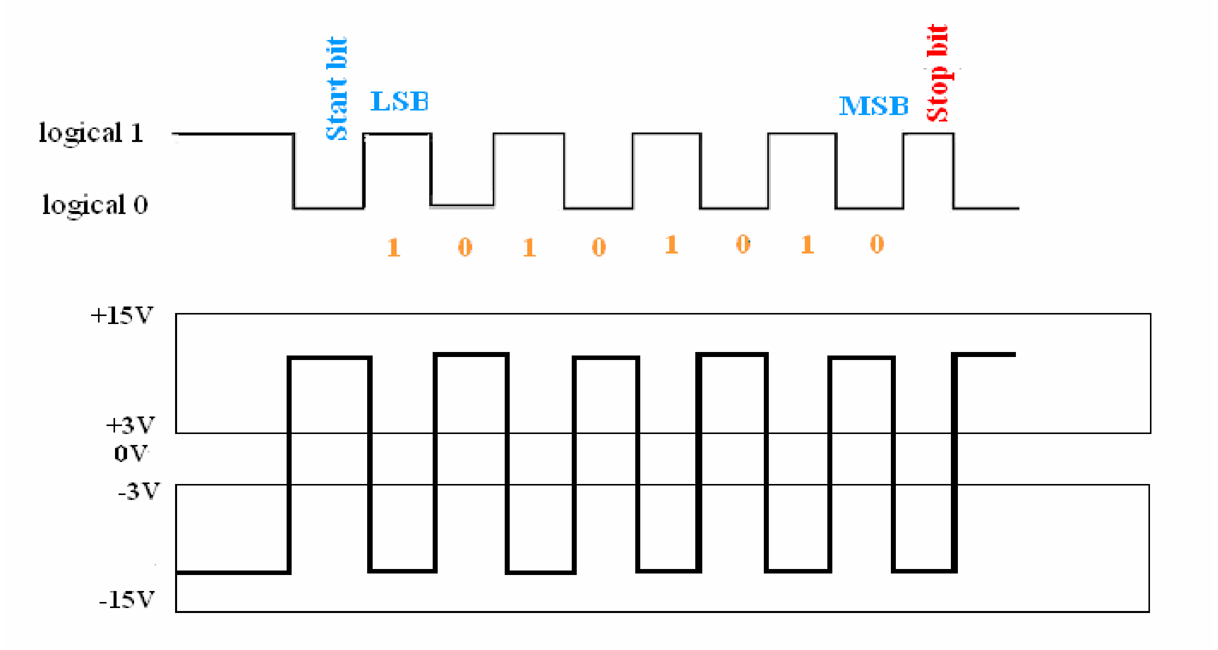


Figure 6.3.5: RS232 Data to microscope

The binary data 01010101 (0x55)   is sent and data observed on the oscilloscope is 101010. Each bit is inverted. This control helps, especially at the beginning to test if the data sent is as well received.

Chapter 7: Conclusion and proposals for further works

7.1 – Conclusion

This project; Hardware and Software for Position Determination and Visualization for an indoor Navigation System has been a success as the task of the thesis has been fully tackled on the part of the hardware design as well as on that of the software . Many mathematical methods in determining the coordinate of an object have been examined. The first examined method was the iterative method (Taylor linearization). This method involved more code in programming. The closed solution method was then examined. This method gave good results and programming was not complicated. A third method was derived from the second method by assuming the area of experiment is rectangular. This means two sides were always the same as the opposite side. It returned good results and also gave the possibility of controlling if the values were in the expected positions. This possibility could be examined by the simple fact that there were many pairs of x and y. A pair of results was compared with another. In case of a mistake, it could be detected even at the level of demonstration on the LED display. This is observed when not only an LED is lighted but two or four.
The choice of the right hardware component was judicious, selecting the right combination chip (IC) and LED matrix display, that of resistors, capacitors and all the others.
The software design has been modular, making it easy for future work, error detection and also rendering it possible to have an eye on the codes while programming. In order to improve on the results (coordinate of the object) some factors have been considered that led to better results. An example is temperature. Reflection was as well taken into consideration in a parallel project.

The PCB design as outlined in chapter four has been progressively developed using Eagle to draw the schematics and finally the board layout. The first board had a few properties which brought the idea of amelioration. One of the main properties was the space between a display module and the next. Though this design, after adaptation was functional, it was decided on a more adequate design. This new design made it possible to produce just one PCB. This helped in avoiding the use of multiple connectors from one display to the next, as well as the errors that could be involved in using these connectors. This second PCB was designed in such a way that the 8 LED matrix display (512 LEDs) could fit on the PCB and the each display fit to the next without a space. This property makes the system as a display technology, more demonstrative.
There are a few measures which could have been considered in improving on the results. Due to some factors like available material and scope of work, these factors were left out. Some of these factors are listed in chapter seven, section 7.2.

7. 2 – Some possible applications of this project

Localization applications have been developed for a number of everyday scenarios. In the industries and research centres, localization can be very important in determining the position of a robot for example, allowing a full control of the robot. Robots are gaining more and more importance nowadays and its control as well.

It could equally be used in museum guides. Many visitors to museums are unfamiliar with the navigation at the location. Having such a display could help such visitors navigate on this unfamiliar place easily. They will need to move around with the display to be able to locate their position.

The developed hardware and software are applicable in research purposes in the IT and automation engineering laboratories of the University of Applied Sciences Hamburg Germany.

In many industries, this project could be applied in logistics in determining the position of a mobile transport cart (carriage) in transporting materials from one angle in the factory to the next. In order to make this realisable, the actual position of this cart is necessary.

7.3 – Proposals for further works

7.3.1 - Outlier detection and corrective measures proposal and analysis

An outlier is an observation that lies outside the overall pattern of a distribution (Moore and McCabe 1999). Usually, the presence of an outlier indicates some sort of problem. This can be a case which does not fit the model under study or an error in measurement. The first step is to detect the outlier. Detection will vary depending on the source. After detection, reduction measures are adapted and if possible substituted.

Due to reflection of sound and other disturbing factors in a room, further measures are taken to assure a better accuracy with the available standard conditions. One of these measures is the detection of outlier and treating them. The diagrams below explain a case study for the four sensors, where three of them indicate similar values and one indicates an extreme value.

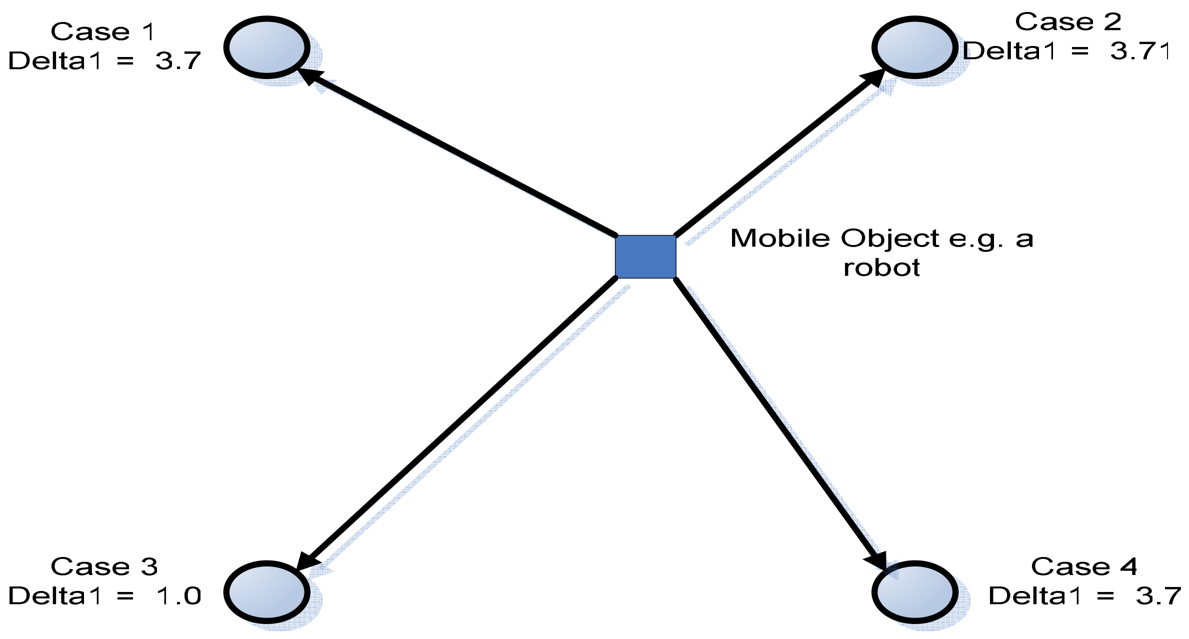


Figure 7.2.1: Setup for outlier detection

Case 1, 2 and 4 are normal and good values but the third case is an outlier. In order to improve on the accuracy, the third values are omitted and calculations done with just the values of the first, second and fourth values. In so doing, the range of accuracy is improved. Mathematically, there is no fixed value for an outlier but ultimately a subjective issue. The cause of an outlier determines how it should be treated. In this case, it is simply left out and it improves on the result as demonstrated on the diagrams above.

Case number	Delta 1 / standardized values
1	3,7
2	3,71
3	1
4	3,7

Table 7.2.2 case study of varied values of Delta 1

A resulting curve from Table 7.2.2 clearly indicates the presence of an outlier. This could equally be deduced from the table.

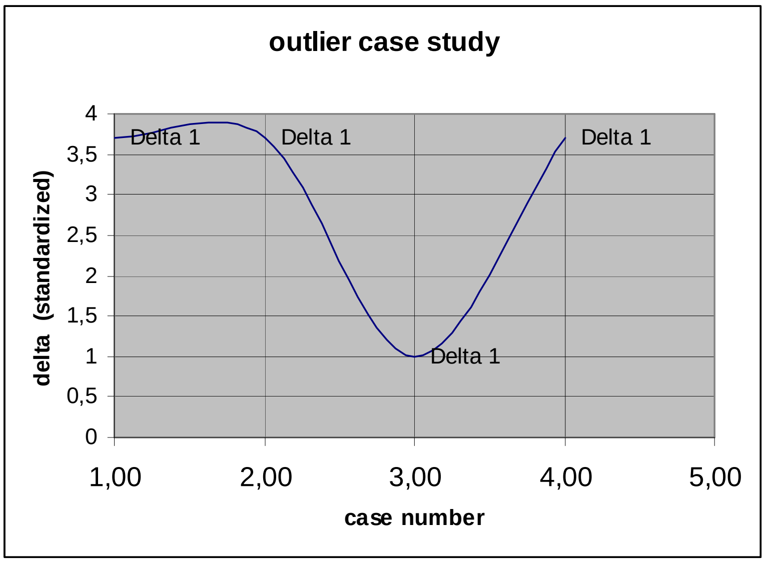


Figure 7.2.2: Outlier case study result before correction

After the outlier detection and correction, the curve is almost a straight line. See the curve below.

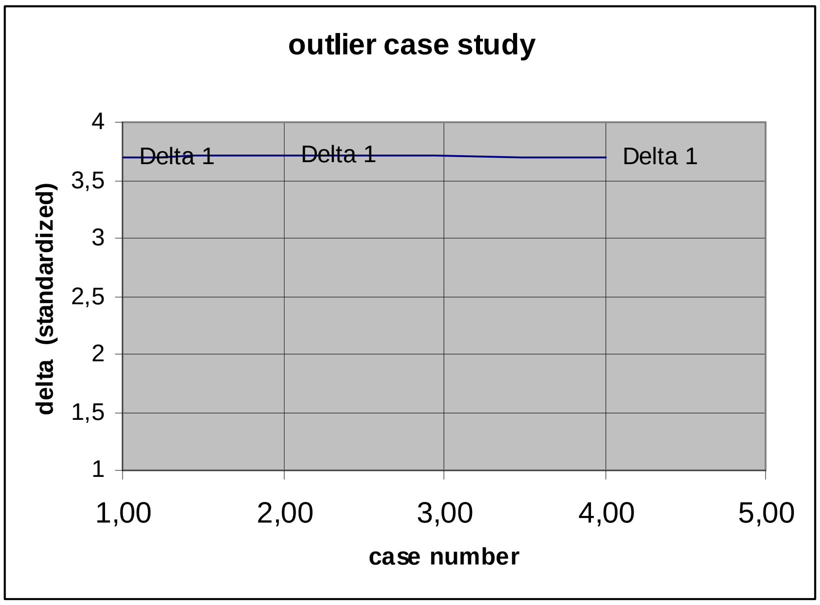


Figure 7.2.3: Outlier case study result after correction

As seen in figure 7.2.2, the results for the various cases are more or less the same where as it there an extreme value (outlier) was lying out of the range. This correction should definitively yield better results if implemented as described. Though better results are expected, it has the disadvantage that few but good data are eliminated. This requires a thorough study of the system and other techniques.

7.3.2 - Kalman filtering as a method for determining the coordinates

Kalman filtering is a relatively recent (1960) development in filtering, although it has its roots as far back as Gauss (1795). Kalman filtering has been applied in areas as diverse as aerospace, marine navigation, nuclear power plant instrumentation, demographic modelling, manufacturing, and many others. The question which is addressed by the Kalman filter is this: Given our knowledge of the behaviour of the system, and given our measurements, what is the best estimate of position and velocity? We know how the system behaves according to the system equation, and we have measurements of the position, so how can we determine the best estimate of the system variables? Surely we can do better than just take each measurement at its face value, especially if we suspect that we have a lot of measurement noise. This aspect could not be verified in the scope of this work but is a point of interest for further works in position determination.

7.3.3 - Using a single voltage source

A further proposal on future works is the use of one voltage source instead of two as it is the case in this project. One supplies the microprocessor MSP430F169STK. It has a voltage of 7.5V. The other source supplies the MAX7221 and LED module, requiring between 3.5V – 5V. The MSP430F169 can not supply this module, since the current requirement of the LEDs can not be met by the microcontroller.

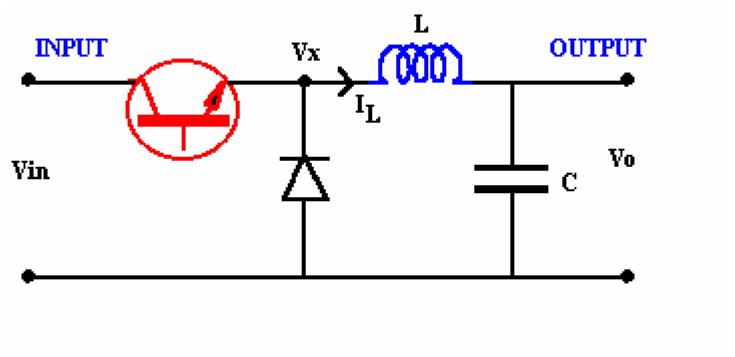


Figure 7.2.4: DC step down converter.

In this circuit the transistor turning ON will put voltage V_{in} on one end of the inductor. This voltage will tend to cause the inductor current to rise. When the transistor is OFF, the current will continue flowing through the inductor but now flowing through the diode. We initially assume that the current through the inductor does not reach zero, thus the voltage at V_x will now be only the voltage across the conducting diode during the full OFF time. The average voltage at Vx will depend on the average ON time of the transistor provided the inductor current is continuous.[31].

$$v_x - v_0 = L\frac{d_i}{d_t}$$

The two voltage sources could be adapted to supply the microcontroller as well as the LEDs by using a step down of the 7.5V at the entrance of the LED module. Typically the output produced is at a different voltage level than the input. In addition, DC-to-DC converters are used to provide noise isolation, power bus regulation.

Chapter 8: -Terminology Bibliography and attachments

8.1 –Terminology

ADC: Analogue-to-Digital Converter

ACLK: Auxiliary Clock
See Basic Clock Module [9]

BOR: Brown-Out Reset

C/A-Code
The standard (Clear/Acquisition) GPS PRN code, also known as the Civilian Code or S-Code. Only modulated on the L1 carrier. Used by the GPS receiver to acquire and decode the L1 satellite signal, and from which the L1 pseudo-range measurement is made.

Coarse Acquisition (C/A)
A spread spectrum direct sequence code that is used primarily by commercial GPS receivers to determine the pseudo-range to a transmitting GPS satellite, modulated on the L1 carrier.

CPU: Central Processing Unit

CTS − Clear To Send
The CTS signal is received from the other end of the serial cable. A space voltage indicates that it is alright to send more serial data from your workstation. CTS is usually used to regulate the flow of serial data from your workstation to the other end.

DAC: Digital-to-Analogue Converter

DCD − Data Carrier Detect
The DCD signal is received from the computer or device on the other end of your serial cable. A space voltage on this signal line indicates that the computer or device is currently connected or on line. DCD is not always used or available.

DCE : Data Circuit-terminating Equipment

DCO: Digitally Controlled Oscillator

DLP
Digital Light Processing (DLP) is a trademark owned by Texas Instruments, representing a technology used in projectors and video projectors.

DRC: Design Rule Checks

Dst: Destination

DTE: Data Terminal Equipment

DTR − Data Terminal Ready
The DTR signal is generated by your workstation and tells the computer or device on the other end that you are ready (a space voltage) or not−ready (a mark voltage). DTR is usually enabled automatically whenever you open the serial interface on the workstation.

FLL: Frequency Locked Loop

GIE: General Interrupt Enable

GND − Logic Ground
Technically the logic ground is not a signal, but without it none of the other signals will operate. Basically, the logic ground acts as a reference voltage so that the electronics know which voltages are positive or negative.

GPS General Positioning System

IEEE
Institute of **E**lectrical and **E**lectronics **E**ngineers. IEEE is one of the leading standards-making organizations in the world. IEEE performs its standards making and maintaining functions through the IEEE Standards Association (IEEE-SA). IEEE standards affect a wide range of industries including: power and energy, biomedical and healthcare, Information Technology (IT) [19].

INT(N/2): Integer portion of N/2

Intensity
Intensity **(mcd)** stands for Milli Candle Power and measures the intensity of LED.

Interferometry
Interferometry is the technique of superimposing (interfering) two or more waves, to detect differences between them.

Interrupt
An interrupt is an event in hardware that triggers the processor to jump from its current program counter to a specific point in the code. Interrupts are designed to be special events whose occurrence cannot be predicted precisely (or at all). The MSP has many different kinds of events that can trigger interrupts, and for each one the processor will send the execution to a specific point in memory.

I/O: Input/Output

ISR: Interrupt Service Routine

JTAG
Joint Test Action Group [21]

LCD
A liquid crystal display (LCD) is a thin, flat display device made up of any number of colour or monochrome pixels arrayed in front of a light source or reflector. It is often utilized in battery-powered electronic devices because it uses very small amounts of electric power.

LcoS
Liquid crystal on silicon (LCOS or LCoS) is a "micro-projection" or "micro-display" technology typically applied in projection televisions-.

LPS Local Positioning System

LPM: Low-Power Mode

LSB: Least-Significant Bit

LSD: Least-Significant Digit

MAB: Memory Address Bus

MCLK: Master Clock
See Basic Clock Module [9]

MDB: Memory Data Bus

MSB: Most-Significant Bit

MSD: Most-Significant Digit

NMI: (Non)-Maskable Interrupt
See System Resets Interrupts and Operating Modes [9]

PC: Program Counter
See RISC 16-Bit CPU

P-Code
The Precise or Protected code. A very long sequence of PRN binary biphase modulations on the GPS L1 and L2 carrier at a chip rate of 10.23MHz, which repeats about every 267 days. Each one week segment of this code is unique to a GPS satellite and is reset each week. Under the policy of "Anti-Spoofing" the US Dept. of Defense has encrypted the P-Code (replacing it with a so-called Y-Code). Only US military and other authorised users are able to overcome AS using special receivers.

See System Resets Interrupts and Operating Modes [9]

PUC: Power-Up Clear

Pseudo-Random Noise (PRN)
A binary signal with random noise-like properties. It is generated by mathematical algorithm or "code", and consists of repeated pattern of 1's and 0's. This binary code can be modulated on the GPS carrier waves using Binary Shift-Key (BSK) modulation. The C/A-Code and the P-Code are examples of PRN codes. Each satellite transmits a unique C/A-Code and P-Code sequence (on the same L1 and L2 frequencies), and hence a satellite may be identified according to its "PRN number", e.g. PRN2 or PRN14 are particular GPS satellites.

Pseudo-Range
A distance measurement based on the correlation of a satellite's transmitted code (may be the C/A-Code or the encrypted P-Code) and the local receiver's reference code (for that PRN satellite number), that has not been corrected for errors in synchronisation between the transmitter's clock and the receiver's clock. Hence a pseudo-range measurement is a time-error biased distance measurement. The precision of the measurement is a function of the resolution of the code; hence C/A-Code pseudo-range measurements may have a "noise" at the few metre level for standard GPS receivers (and at the sub-metre precision level in the case of so-called "narrow correlator" GPS receivers). [20]

PUC: Power-Up Clear
See System Resets Interrupts and Operating Modes [9]

QSPI:
Queued serial peripheral interface (QSPI)

RAM: Random Access Memory

RISC:
The acronym RISC (pronounced risk), for reduced instruction set computing, represents a CPU design strategy emphasizing the insight that simplified instructions which "do less" may still provide for higher performance if this simplicity can be utilized to make instructions execute very quickly.[30]

RS-232: Recommended Standard 232

RTS − Request To Send
The RTS signal is set to the space voltage by your workstation to indicate that more data is ready to be sent. Like CTS, RTS helps to regulate the flow of data between your workstation and the computer or device on the other end of the serial cable. Most workstations leave this signal set to the space voltage all the time.

Run time differences
In the context of this project, the run time differences are the time at which a sensor (microphone) receives the sound signal in comparison with another.

RXD − Received Data
The RXD signal carries data transmitted from the computer or device on the other end to your workstation. Like TXD, mark and space voltages are interpreted as 1 and 0, respectively.

SCG: System Clock Generator
See System Resets Interrupts and Operating Modes

SHT: Sample and hold time

SFR: Special Function Register

SMCLK: Sub-System Master Clock
See Basic Clock Module

SP: Stack Pointer. See RISC 16-Bit CPU

SR: Status Register

Src: Source . See RISC 16-Bit CPU

SPI : Serial peripheral interface

TOS: Top-of-Stack See RISC 16-Bit CPU

TXD − Transmitted Data
The TXD signal carries data transmitted from your workstation to the computer or device on the other end (like a MODEM). A mark voltage is interpreted as a value of 1, while a space voltage is interpreted as a value of 0.

UART
A universal asynchronous receiver/transmitter

USART
A universal synchronous and asynchronous receiver/transmitter

UTXEx: Transmit enable bit

UxRXBUF: Receive buffer

UxTXBUF: Transmit buffer

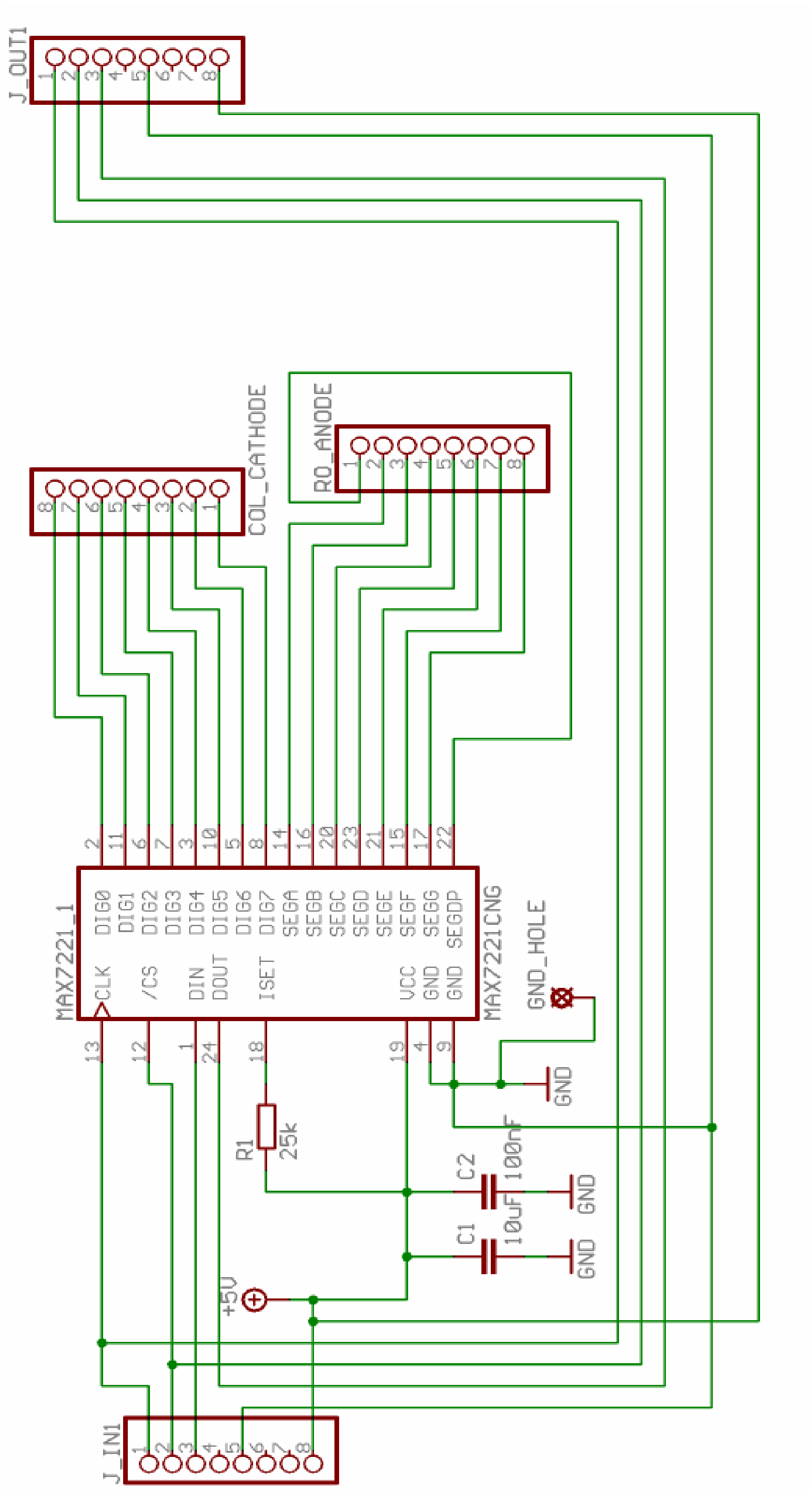
Wafer
A wafer is a thin slice of semiconductor material, such as a silicon crystal, used in the fabrication of integrated circuit and other micro devices.

WDT: Watchdog Timer.

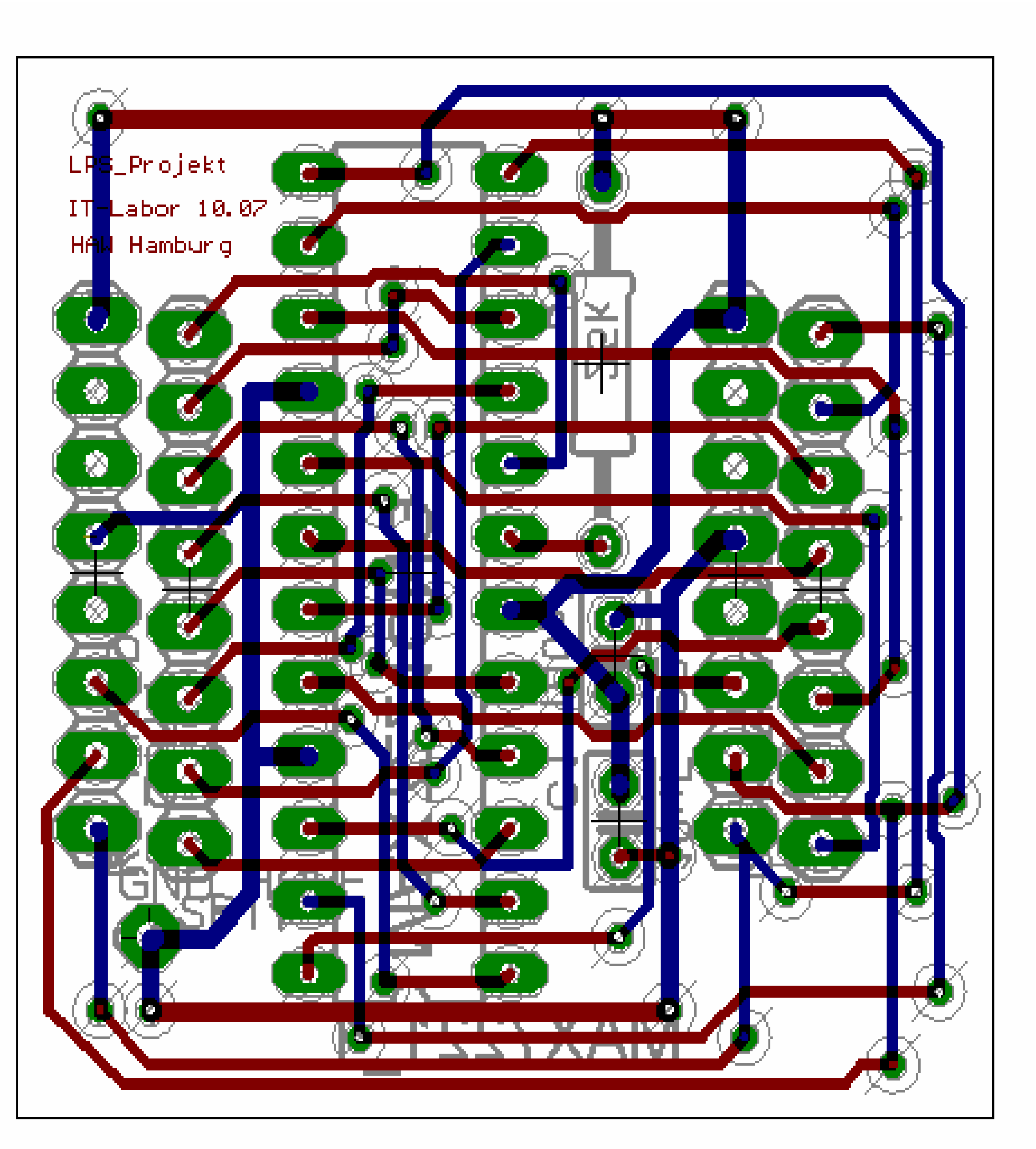
8.2 - Bibliography

[1] Grundlagen und Anwendung globaler Satellitennavigationssystem 2. Edition by Prof. Dr. Ing. habil Werner Mansfeld
[2] Journal ; Elektonik und Informationstechnik (e&i) Third issue of 3.2008
[3] http://www.gnu.org/software/gdb/ as of 17.04.08 at 12:10
[4] Richard M. Stallman and Bjrn Remset 1987
Richard Stallman lecture at the Royal Institute of Technology, Sweden (1986-10-30)
[5] Lecture material of Lecture MC and Lab MCL of Prof. Dr. Ing Riemschneider
University of Applied Sciences Hamburg Germany
[6] http://www.network-theory.co.uk/docs/gccintro/gccintro_4.html :11:52 17 April 2008
[7] The Master Handbook of Acoustics. New York: McGraw-Hill)
[8] Embedded System Design Using the TI MSP430 Series by Chris Nagy, Published by Elsevier Science (USA) 003
[9] MSP430F169 Manual from Texas Instruments
[10 Source: Journal of Basic Engineering, Vol. 82: pp. 35-45. and from Lange, A. 1999. Finnish Meteorological Institute Contributions, No. 22, Helsinki, Finland
[11] http://www.lumileds.com as of 17.04.08
[12] http://www.imagehouse.dk/default.asp?file=products_lightsources_advantagesof.htm as of 12.05.08
[13] http://dictionary.die.net/jtag as of 19.04.2008 at 10:09:01 AM
[14] http://cnx.org/ 15 as of 19.04.2008 at 11:20:03 AM
[16] http://www.kowoma.de/en/gps/history.htm as of Monday, 24 March 2008 at 06:45
[17] http://myhome.spu.edu/bolding/EE4211/EagleTutorial4.htm as of 13.04.08
[18] http://www.cadsoftusa.com/ as of 13.04.08
[19] http://standards.ieee.org/ as of 19/04/2008 12:36
[20] http://www.gmat.unsw.edu.au/snap/gps/glossary_r-z.htm
[21] http://www.techweb.com/encyclopedia/defineterm.jhtml?term=JTAG as of 19/04/2008 12:32
[22] Some Experiments on Colours, Nature **111**, 1871, in John William Strutt (Lord Rayleigh) (1899). Scientific Papers. University Press.
[23] http://www.network-theory.co.uk/docs/gccintro/gccintro_4.html :11:52 17 April 2008
[24] Lecture material Richard M. Stallman and Bjrn Remset Richard Stallman lecture at the Royal Institute of Technology, Sweden (1986-10-30)
[25] Physics by Prof Dr. Tao Pang Cambridge University Press, Cambridge, UK, 2006)
Publisher: Cambridge University Press
Published: February 2006
[26] http://www.gnu.org/software/binutils/ as of 17.04.08 12:36)
[27] The Master Handbook of Acoustics. New York: McGraw-Hill
[28] Microcontroller Programming The microchip PIC by Julio Sanchez Minnesota State University, Mankato and Maria P. Canton, South Central College, North Mankato, Minnesota 2007
[29] EVERLIGHT LED Data Sheet
(http://www.datasheetcatalog.com/everlightelectronics/1/) as of 21.11.2007
[30] http://en.wikipedia.org/wiki/Reduced_instruction_set_computer as of 04.06.2008 at 22:00
[31] http://www.powerdesigners.com/InfoWeb/design_center/articles/DC-DC/converter.shtm as of 20.06.2008
[32] http://ww.dansdata.com/images/caselight/friodea380.jpg as of 12.03.2008 at 18:34

8.3 - Attachments



Attachment 1: Schematics of first PCB design version



Attachment 2: Board of first PCB design

Attachment 3 and 4 are the schematics and board of the second and final design of PCB and are attached as extra pages at the end of this document.

A3 format : Attachment 3: Schematics of second PCB design

A3 format: Attachment 4:   Board of second PCB design

Content of the CD

Folder1: Hardware Development

-pin list of second PCB design
-part list of second PCB design
-net list of second PCB design
-some pictures of hardware

Folder2: LPS_test

-various C codes

Folder3: Matlab files

Folder4: Documentation in PDF format

Folder5: Excel files

Folder6: Files from Eagle

Versicherung über die Selbständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §25(4) ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen Kenntlich gemacht.

Ort Datum Unterschrift