Mansoura University
Faculty of Engineering
Mechatronics Engineering Program

Shower Water Monitor

(Training 1 Project)

Under supervision of:

Eng. Abdelrahman Omar
Eng. Ahmed Ramadan

Prepared by:

Alaa Mohamed Nasser Mohamed Ibrahim Elsayed
Ahmed Ismail Osman Eldesoki
Ahmed Hesham Ahmed Elzehery

2021-2022

TABLE OF CONTENTS

Team members

1- **Name:** Alaa Mohamed Nasser Mohamed Ibrahim Elsayed

ID: 800136619

2- **Name:** Ahmed Ismail Osman Eldesoki

ID: 800136431

3- **Name:** Ahmed Hesham Ahmed Elzehery

ID: 800136490

Abstract

As we know water plays an important role in our life so we will talk about the importance of saving it.

After thinking for a long time, we found that most of wasted water when we use the shower.

As well as we found that different shower heads can use anywhere from 9.5 litres (2.5 gallons) per minute to less than 6 litres (1.6 gallons) per minute.

Furthermore, Egypt has a water poverty problem and the person's intake of water is less than the average, so we decided to give a little help to make people aware of our high-water usage.

So, we intended to design and build a device that would work on different aspects of our daily use of water in shower. It can display the total volume of water used per shower, the cost of the water, and the flow rate.

In spite of the capability of that device, we have provided an additional feature which will alarm the person that he/she has exceeded the average amount of water per shower (65.1 litres).

We have made this device to be compatible with everyday use starting from making it waterproof to make it rechargeable.

We've had this device installed for a few weeks and it's handy to have a live readout of the amount of water being used.

Chapter (1):

Working Principle

As we know from the introduction that our device used to measure the total volume of water used per shower, the cost of the water, and the flow rate.

Our device depends on the signal that is generated from the Water Flow Meter Sensor (YF-S201). As this sensor generate the pulses which is directly proportional to the volume of the water.

We receive the pulses from the sensor through our PCB then we transmit these pulses to our AVR Processor which process this signal and produce a number which indicates the total volume of the water after that we do some mathematical calculation to get the flow rate and cost of water.

In the further step we need to make these values (volume, flow rate, and cost) visible to the user, therefore we used LCD 2*16 to display the values on it as follows:
1st mode It can display the total volume and flow rate
2nd mode It displays the total volume and the cost of used water

For a better work in everyday use we have added some features which shall make the experience of using such device achieves the difficult equation that is to be industrial and suitable for personal use.

First of all, we had to make it rechargeable for portability and recurrent use. So, we have provided it with Lithium-ion battery and a circuit for charging it which consist of charging board and boost converter.

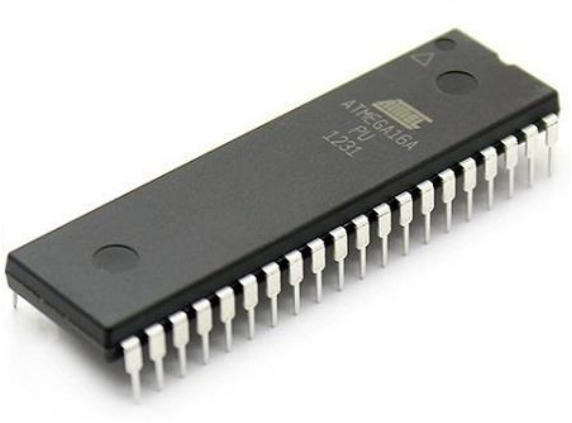
The other feature is to make completely waterproof, therefore it can be used beside a shower without any doubt that it may be damage or not.

Chapter (2):

Hardware

2.1 Hardware Components

2.1.1 ATmega16



Features:

• High-performance, Low-power Atmel® AVR® 8-bit Microcontroller

• Advanced RISC Architecture

• High Endurance Non-volatile Memory segments

I. 16K Bytes of In-System Self-programmable Flash program memory
II. 512 Bytes EEPROM
III. 1K Byte Internal SRAM

• Peripheral Features

I. Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
II. One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
III. Real Time Counter with Separate Oscillator
IV. Four PWM Channels

• Special Microcontroller Features

I. Power-on Reset and Programmable Brown-out Detection
II. Internal Calibrated RC Oscillator
III. External and Internal Interrupt Sources

• I/O and Packages

I. 32 Programmable I/O Lines

• Operating Voltages

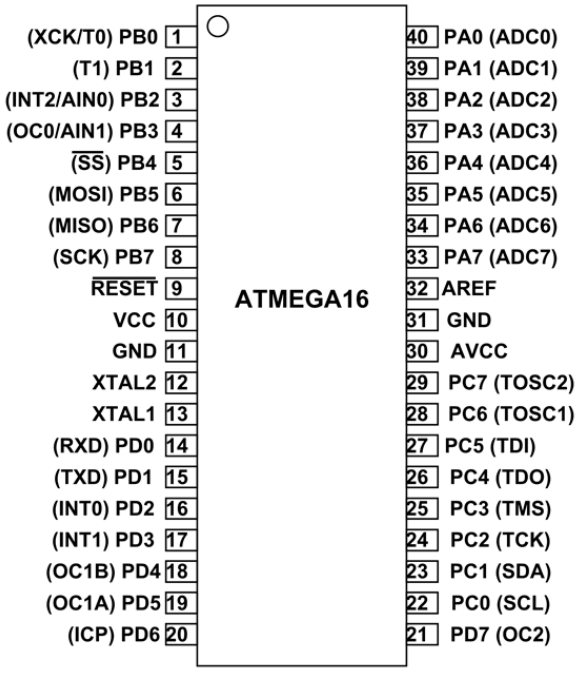
I. 2.7 - 5.5V for ATmega16L

• Speed Grades

I. 0-8 MHz for ATmega16L

• Power Consumption @ 1 MHz, 3V, and 25·C for ATmega16L

I. Active: 1.1 mA
II. Idle Mode: 0.35 mA
III. Power-down Mode: < 1 μA



2.1.2 Flow Sensor

Description:

Water flow sensor consists of a plastic valve from which water can pass. A water rotor along with a hall effect sensor is present the sense and measure the water flow.

When water flows through the valve it rotates the rotor. By this, the change can be observed in the speed of the motor. This change is calculated as output as a pulse signal by the hall effect sensor. Thus, the rate of flow of water can be measured.

Working Principle:

The main working principle behind the working of this sensor is the Hall effect. According to this principle, in this sensor, a voltage difference is induced in the conductor due to the rotation of the rotor. This induced voltage difference is transverse to the electric current.

When the moving fan is rotated due to the flow of water, it rotates the rotor which induces the voltage. This induced voltage is measured by the hall effect sensor and displayed on the LCD display.

Technical Parameters:

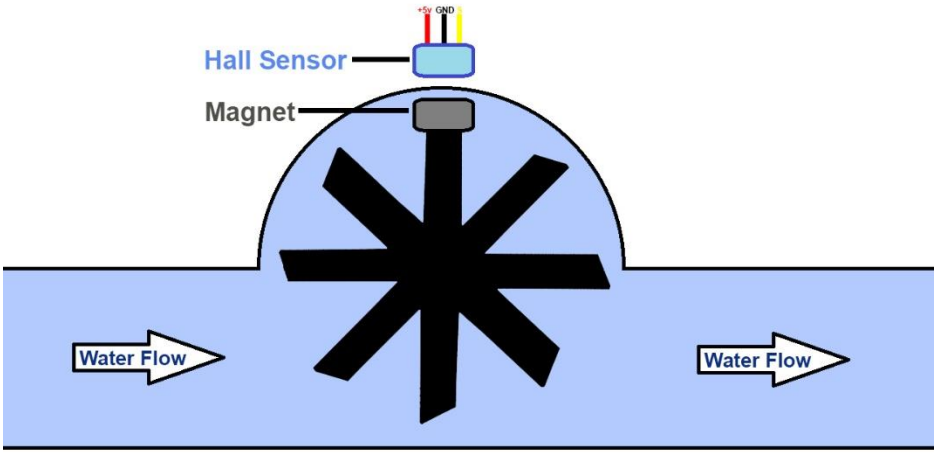
- the minimum rated operating voltage DC 5V-24V
- the maximum operating current of 15 mA (DC 5V)
- the working voltage range DC 5 ~ 18V
- the load capacity of ≤ 10 mA (DC 5V)
- the use of temperature range ≤ 80 ℃
- the use of humidity range of 35% to 90% RH (no frost state)
- to allow pressure water pressure below 1.75Mpa
- save the temperature -25 ~ +80 ℃
- save humidity 25% ~ 95% RH

Specifications:

- Working Voltage: DC 4.5V~24V
- Normal Voltage: DC 5V~18V
- Max. Working Current: 15mA (DC 5V)
- Load capacity: ≤ 10 mA (DC 5V)
- Flow Rate Range: 1~30L/min
- Load Capacity: ≤10mA (DC 5V)
- Operating Temperature: ≤80℃
- Liquid Temperature: ≤120℃
- Operating Humidity: 35%〜90%RH
- Allowing Pressure: ≤1.75MPa
- Storage Temperature: -25〜+ 80℃
- Storage Humidity: 25%〜95%RH
- Electric strength 1250V/min
- Insulation resistance ≥ 100MΩ
- External threads: 1/2"
- Outer diameter: 20mm
- Intake diameter: 9mm
- Outlet diameter: 12mm

Features:

- Model: YF-S201
- Sensor Type: Hall effect
- Working Voltage: 5 to 18V DC (min tested working voltage 4.5V)
- Max current draw: 15mA @ 5V
- Output Type: 5V TTL
- Working Flow Rate: 1 to 30 Litres/Minute
- Working Temperature range: -25 to +80°C
- Accuracy: ±10%
- Maximum water pressure: 2.0 MPa
- Output duty cycle: 50% +-10%
- Output rise time: 0.04us
- Output fall time: 0.18us
- Flow rate pulse characteristics: Frequency (Hz) = 7.5 * Flow rate
- Pulses per Litre: 450
- Durability: minimum 300,000 cycles
- 1/2" nominal pipe connections
- 0.78" outer diameter
- 1/2" of thread
- Size: 2.5" x 1.4" x 1.4"



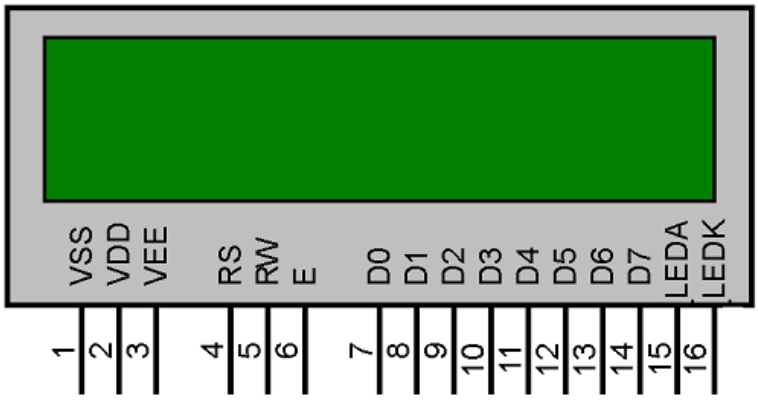
2.1.3 LCD

Description:

A liquid crystal display or LCD draws its definition from its name itself. It is a combination of two states of matter, the solid and the liquid. LCD uses a liquid crystal to produce a visible image.

Working Principle:

The principle behind the LCDs is that when an electrical current is applied to the liquid crystal molecule, the molecule tends to untwist. This causes the angle of light which is passing through the molecule of the polarized glass and causes a change in the angle of the top polarizing filter. As a result, a little light is allowed to pass the polarized glass through a particular area of the LCD.



Connection details:

Pin1 (Ground/Source Pin): This is a GND pin of display, used to connect the GND terminal of the microcontroller unit or power source.

Pin2 (VCC/Source Pin): This is the voltage supply pin of the display, used to connect the supply pin of the power source.

Pin3 (V0/VEE/Control Pin): This pin regulates the difference of the display, used to connect a changeable POT that can supply 0 to 5V.

Pin4 (Register Select/Control Pin): This pin toggles among command or data register, used to connect a microcontroller unit pin and obtains either 0 or 1(0 = data mode, and 1 = command mode).

Pin5 (Read/Write/Control Pin): This pin toggles the display among the read or writes operation, and it is connected to a microcontroller unit pin to get either 0 or 1 (0 = Write Operation, and 1 = Read Operation).

Pin 6 (Enable/Control Pin): This pin should be held high to execute Read/Write process, and it is connected to the microcontroller unit & constantly held high.

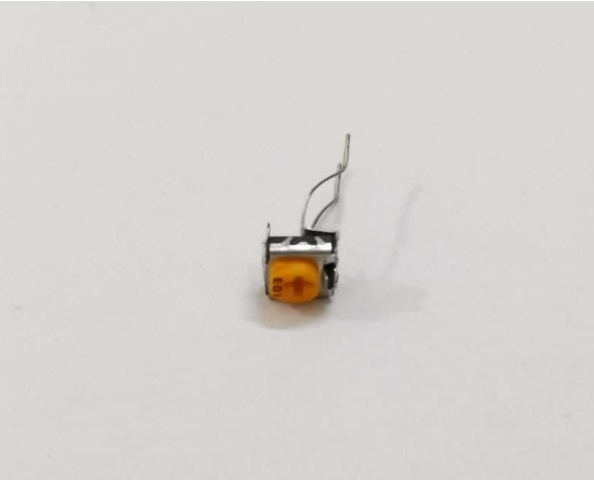
Pins 7-14 (Data Pins): These pins are used to send data to the display. These pins are connected in two-wire modes like 4-wire mode and 8-wire mode. In 4-wire mode, only four pins are connected to the microcontroller unit like 0 to 3, whereas in 8-wire mode, 8-pins are connected to microcontroller unit like 0 to 7.

Pin15 (+ve pin of the LED): This pin is connected to +5V

Pin16 (-ve pin of the LED): This pin is connected to GND.

2.1.4 10k Ohm Potentiometer

Description:



Potentiometer or pot is a three-terminal variable resistor. Resistor, a bundle of resistance, is one of the commonly used components in an electric circuit. As the names suggest, a fixed resistor has a fixed value of resistance, whereas a variable resistor possesses resistance value over a defined range.

Working Principle:

A potentiometer is a simple knob that provides a variable resistance from its value we can control the rate at which an LED blinks which affect the total brightness of the LCD.

2.1.5 (3.5mm) Male Jack

Description & Features:

Stereo Mini Audio Cable 3.5mm Male to 3.5mm Male (1.5m) and we cut it into half and used each half, one for the powering circuit and the other for the sensor output.

2.1.6 Battery

Working Principle:

A battery is made up of an anode, cathode, separator, electrolyte, and two current collectors (positive and negative). The anode and cathode store the lithium. The electrolyte carries positively charged lithium ions from the anode to the cathode and vice versa through the separator. The movement of the lithium ions creates free electrons in the anode which creates a charge at the positive current collector. The electrical current then flows from the current collector through a device being powered to the negative current collector. The separator blocks the flow of electrons inside the battery.

Charge / Discharge Operation:

While the battery is discharging and providing an electric current, the anode releases lithium ions to the cathode, generating a flow of electrons from one side to the other. When plugging in the device, the opposite happens: Lithium ions are released by the cathode and received by the anode.

Technical Parameters:

- Output ranges from 4.2V when completely charged to 3.7V.
- This battery has a capacity of 2000mAh.

2.1.7 Buzzer

Description:

An audio signalling device like a beeper or buzzer may be electromechanical or piezoelectric or mechanical type. The main function of this is to convert the signal from audio to sound. Generally, it is powered through DC voltage.

Working Principle:

When a potential difference is applied across these crystals, they push one conductor and pull the other conductor by their internal property. The continuous pull and push action generate a sharp sound wave.

Features:

1. Operation Voltage: 3-24V DC
2. Current: <15mA
3. SPL: 85dBA/10cm
4. Frequency: 3,300Hz
5. Colour: Black
6. Operating Temperature: – 20° to +60°C

2.1.8 Push Button

Working Principle:

Pressure is placed on the button or actuator, resulting in the depression of the internal spring and contacts and the touching of stable contacts at the bottom of the switch. This process will either close or open the electrical circuit. The repeat application of pressure will cause the spring to retract and alter the status of the push button connection.

Features:

- 12 x 12 Standard Tact Switch
- Stem size 4mm
- Operating life: 100K cycles
- Operating Force: 100, 130, 160 & 250 gf.

2.1.9 AC Switch (ON/OFF Power)

Description:

SPST ON/OFF rocker switch AC 250V/6A – 125V/10A

This type can be used to switch the power supply to a circuit.

2.1.10 Waterproof Button

Features:

- Metal self-locking push button
- Hole size: 12mm
- Stainless steel plated brass
- Install Diameter φ12mm
- Tail configuration connection terminal (2.8×0.5)
- switch specifications 2A/36VDC
- Contact resistance ≤50mΩ
- Insulation resistance ≥1000mΩ
- Ambient temperature: -20°C ~ +55°C
- Mechanical life > 500,000 cycles
- Electrical life > 50,000 cycles
- Panel thickness: 1~ 6 mm
- Nut torque: 5 ~ 14 N.m

- Operation pressure: about 5N
- Operation stroke: about 2.0mm
- Protection level: IP67, IK08
- Material for contact: silver alloy
- Material for button: Stainless steel/Nickel plated brass
- Material for enclosure: Stainless steel/Nickel plated brass
- Material for base: PA66
- LED type: Ring illuminated (LED)
- LED rated voltage: 12V
- LED colour: Red
- LED life: 40000hours

2.1.11 IDC10 Pin Header

Features:

- Polarized IDC10 Male Header Used with Polarized IDC Socket Connectors
- Gold Plated Contacts
- Standard .025″ square pins on .100″ x .100″ grid
- Current Rating Up To 1A
- Ideal for Port Bus and LCD Connections

2.1.12 Charging Board

Description:

A lithium battery charging board is a module that is used for charging lithium batteries. The term board is coined because the charging circuit is assembled on a board made of wood or some insulating material for protection. The charger breakout board is mostly based on the TP4056 circuit. It uses a micro-USB for connection between the breakout board and the computer/USB wall adapter. TP4056 is a constant voltage/current linear charger for single-cell lithium-ion batteries.

Features:

- Charge module: Linear charging
- Current: 1A adjustable
- Charge precision: 1.5%
- Input voltage: 4.5V-5.5V
- Full charge voltage: 4.2V
- Led indicator: red is charging Green is full charged
- Input interface: micro-USB
- Work temperature: -10 C to +85 C
- Inversed polarity: NO
- Size: (Small) 25*19*10mm

2.1.13 Boost Converter

Description:

A boost converter is one of the simplest types of the switch-mode converter. As the name suggests, it takes an input voltage and boosts or increases it. All it consists of is an inductor, a semiconductor switch (these days it's a MOSFET since you can get really nice ones these days), a diode, and a capacitor. Also needed is a source of a periodic square wave. This can be something as simple as a 555 timer or even a dedicated SMPS IC like the famous MC34063A IC.

Features:

- AeroSemi MT3608 High Efficiency 1.2MHz 2A Step Up Converter design
- 2A maximum output current
- 2.0 to 24VDC input voltage range
- 28V maximum output voltage (trimpot adjustable)
- 93% peak efficiency (~200mA output current at 5Vin, 12Vout)
- 36 x 17 x 14mm

2.1.14 ZIF Socket

Description:

This is a high-quality, easy to use universal 40 pin ZIF socket that is 0.6" wide. Compatible with 0.3" up to 0.6" wide ICs up to 40-pins. Makes for easy connecting or programming to many DIP ICs.

High conductivity terminals create solid connections. Armature makes it easy to open and close socket.

2.1.15 USB 2.0 Cable

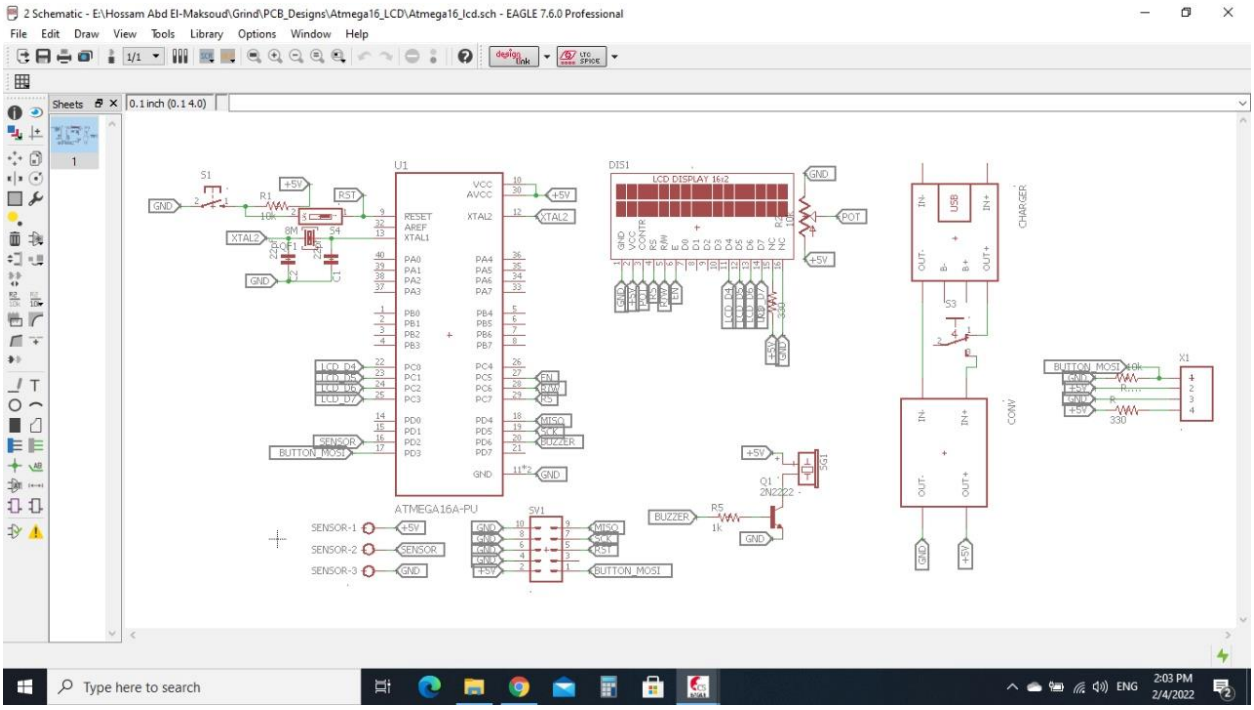
Description:

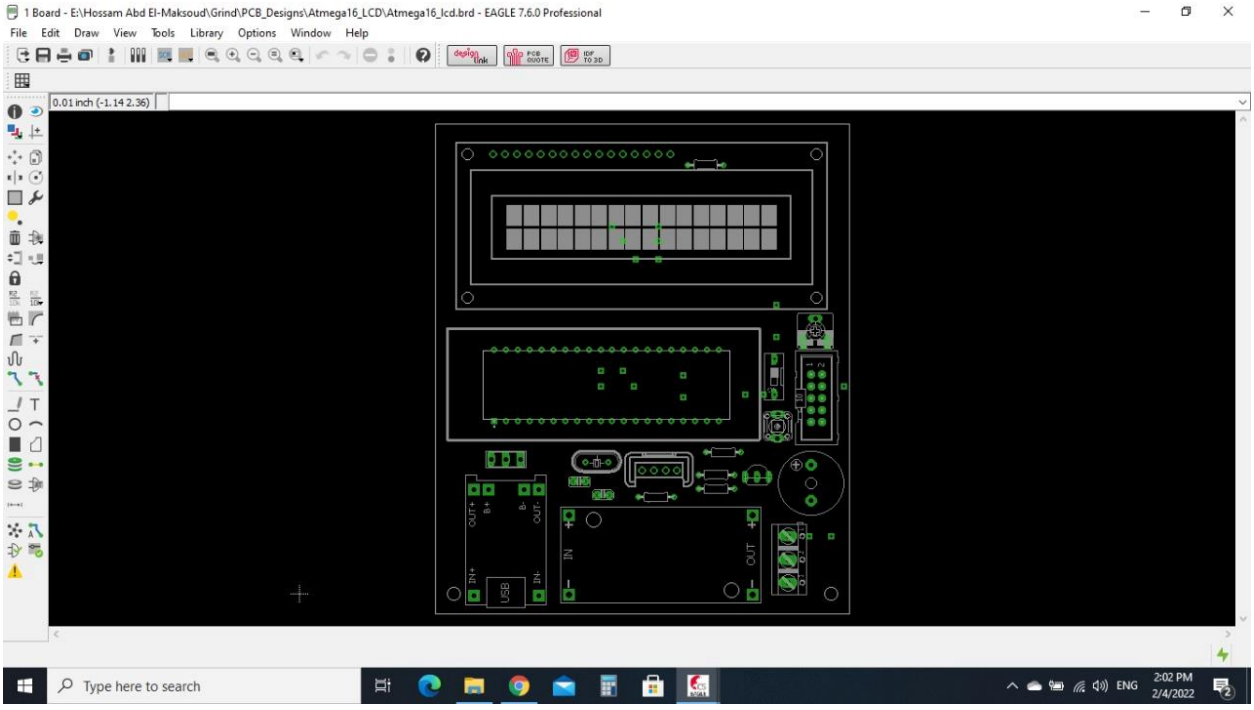
USB 2.0 male to male cable type A (1.5m)

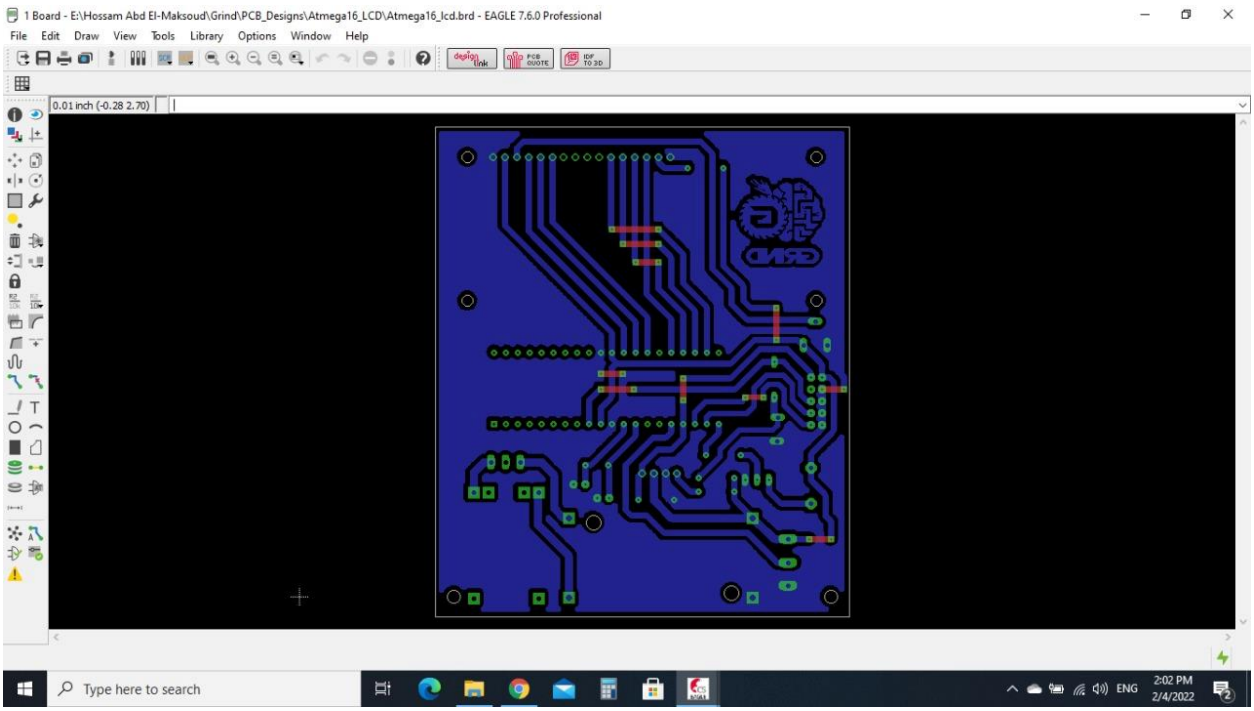
2.1.16 (3.5mm) Female Jack

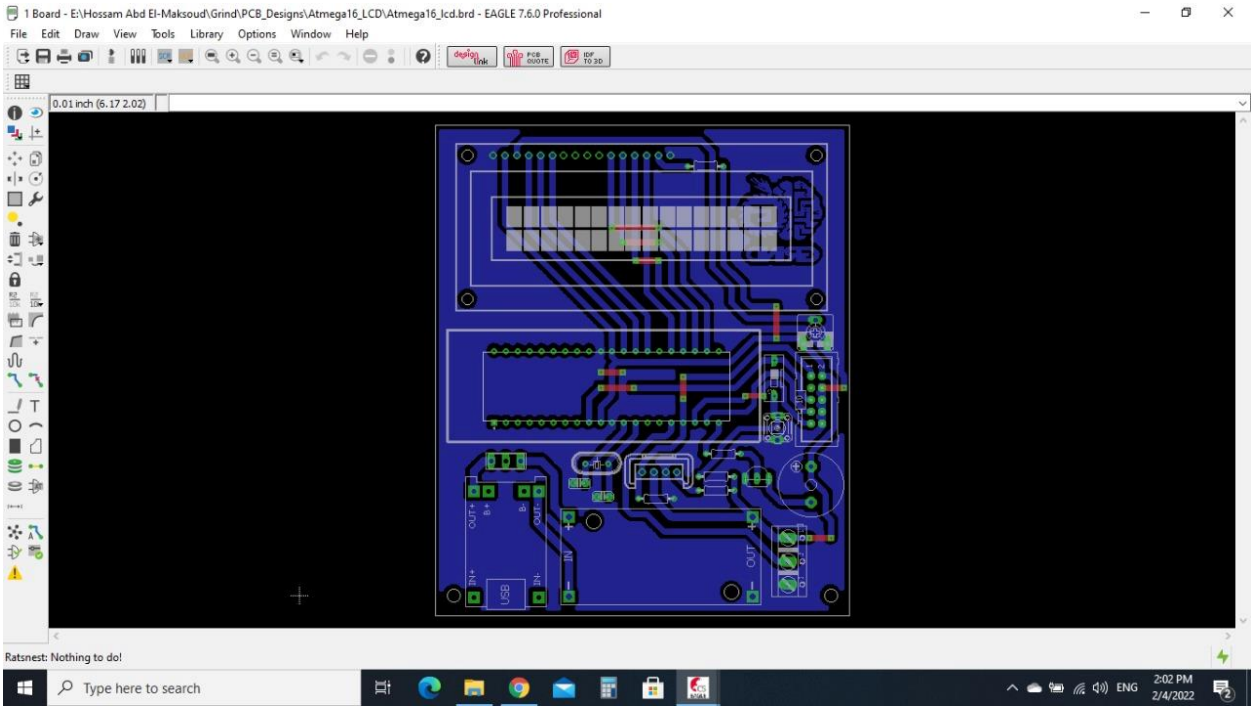
2.1.17 Battery Holder

2.2 PCB DESIGN






1 Board - E:\Hossam Abd El-Maksoud\Grind\PCB_Designs\Atmega16_LCD\Atmega16_lcd.brd - EAGLE 7.6.0 Professional
File Edit Draw View Tools Library Options Window Help
0.01 inch (-0.28 2.70)
Type here to search
ENG
2:02 PM
2/4/2022


1 Board - E:\Hossam Abd El-Maksoud\Grind\PCB_Designs\Atmega16_LCD\Atmega16_lcd.brd - EAGLE 7.6.0 Professional
File Edit Draw View Tools Library Options Window Help
0.01 inch (6.17 2.02)
Ratsnest: Nothing to do!
Type here to search
ENG
2:02 PM
2/4/2022

Chapter (3):

Design

3.1 Waterproofing

One of the most difficult aspects of this project is making the whole thing waterproof. Since it will reside in a shower, it must be able to survive extreme humidity and the occasional splash.

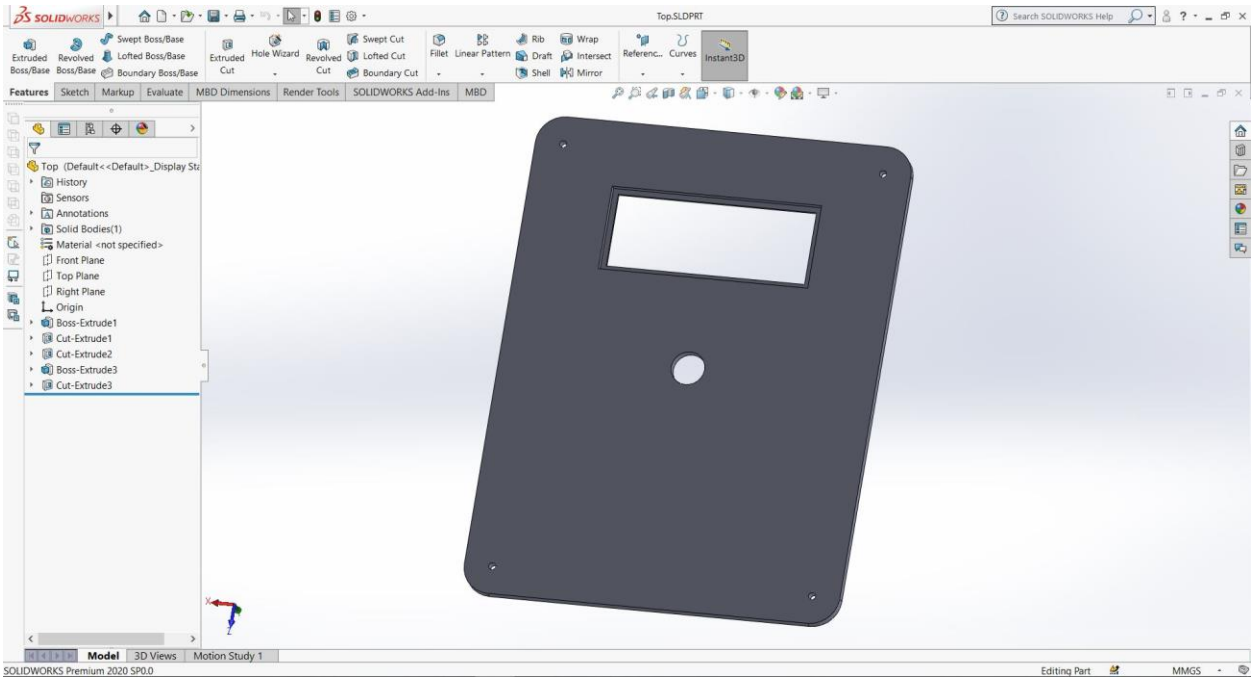
We have decided to make our enclosure 3D printed to fit the waterproofing requirements.

We have used PLA (Polylactic acid) to print the enclosure. It is one of the most popular materials used in 3D printing. It is the default filament of choice for most extrusion-based 3D printers because it can be printed at a low temperature and does not require a heated bed. PLA is a great first material to use to create parts that can be used for a wide variety of applications. It is also one of the most environmentally friendly filaments on the market today. Derived from crops such as corn and sugarcane, PLA is renewable and most importantly biodegradable. As a bonus, this also allows the plastic to give off a sweet aroma during printing.

3.2 CAD

The enclosure consists of two parts, a box and a cover. It was completely designed on SolidWorks.

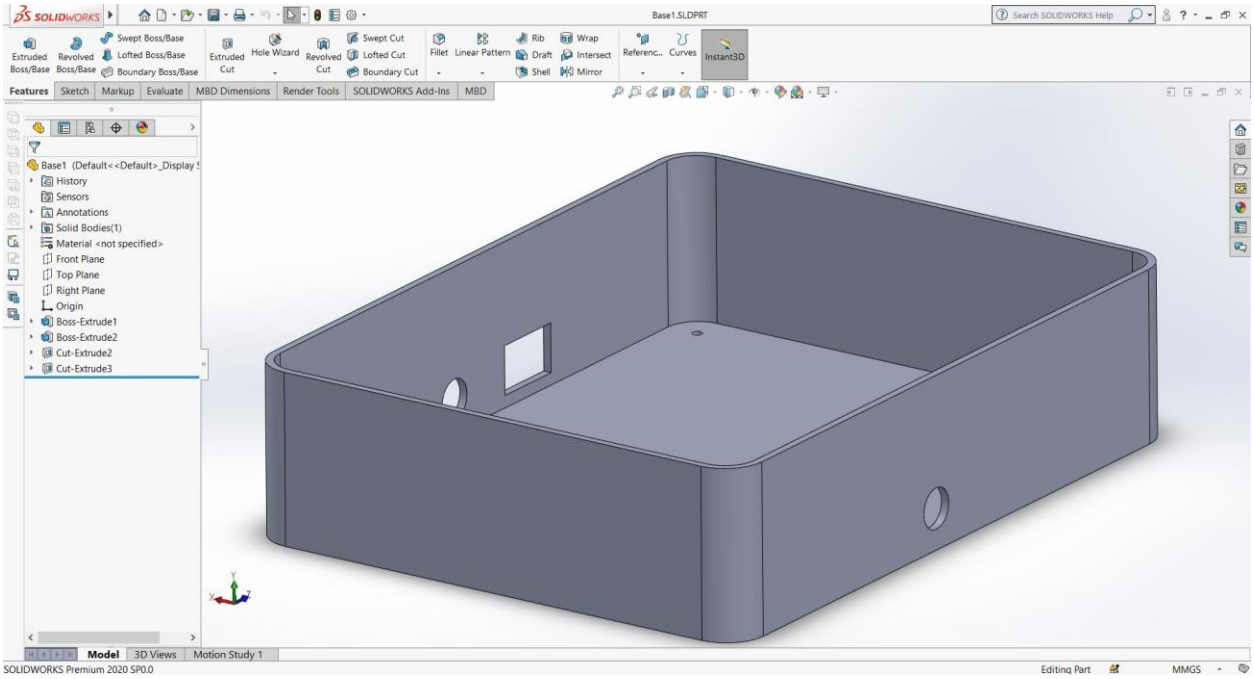
3.2.1 Cover



Specifications:

- Dimensions: 175mm*135mm*8.5mm
- Thickness: 3.5mm
- LCD dimensions: 71.3mm*26.3mm
- Fillet radius: 10mm
- Waterproof button: 12mm Diameter
- Bolts: M3 screw

3.2.2 Box



Specifications:

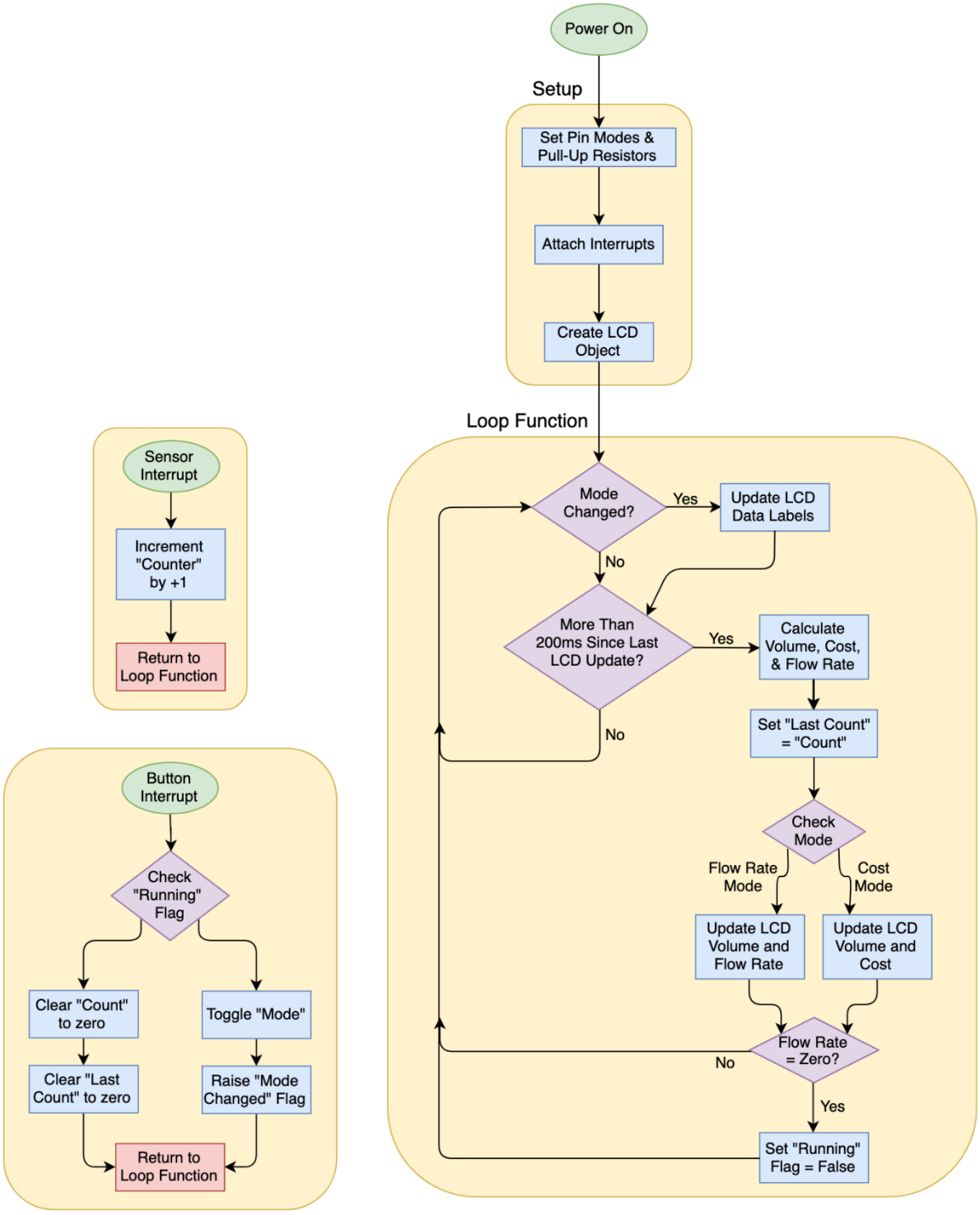
- Dimensions: 175mm*135mm*42mm
- Thickness: 2mm
- Fillet radius: 10mm
- 3.5mm female jack: 10mm Diameter
- AC switch: 18.5mm*12mm
- Bolts: M3 screw

3.3 Final Render

Chapter (4):

Software

4.1 Mind Map


Power On
Setup
Set Pin Modes & Pull-Up Resistors
Attach Interrupts
Create LCD Object
Loop Function
Mode Changed?
Yes
Update LCD Data Labels
No
More Than 200ms Since Last LCD Update?
Yes
Calculate Volume, Cost, & Flow Rate
No
Set "Last Count" = "Count"
Check Mode
Flow Rate Mode
Cost Mode
Update LCD Volume and Flow Rate
Update LCD Volume and Cost
Flow Rate = Zero?
No
Yes
Set "Running" Flag = False
Sensor Interrupt
Increment "Counter" by +1
Return to Loop Function
Button Interrupt
Check "Running" Flag
Clear "Count" to zero
Clear "Last Count" to zero
Toggle "Mode"
Raise "Mode Changed" Flag
Return to Loop Function

4.2 Drivers

4.2.1 LCD

All of LCD pins are on the same port which is **PORTA** of the microcontroller as this port does not have any predefined registers so we can easily set our driver on and it will work properly.

We have chosen to work with 4-bit data mode as it will cost us less pins and this will help us generate a good-looking PCB design as well as we do not need the full speed of the LCD. Therefore, we decided to use the first four pins of the LCD **from PIN (0) to PIN (3)**.

The other hardware pins are connected and programmed as follows:

- Enable Pin is connected to PIN (5) of the Microcontroller
- Read/Write is connected to PIN (6) of the Microcontroller
- Register Select is connected to PIN (6) of the Microcontroller

Then, we have implemented various functions to be generic for every project in the future and to be portable to work with any IDE.

The function names and description are implemented in that way:

- **LCD_init:** Initialize the LCD by setting up the pins direction and choosing the data mode to work with.
- **LCD_sendCommand:** Send the required command to the screen.
- **LCD_displayCharacter:** Display the required character on the screen
- **LCD_displayString:** Display the required string on the screen
- **LCD_moveCursor:** Move the cursor to a specified row and column index on the screen
- **LCD_displayStringRowColumn:** Display the required string in a specified row and column index on the screen
- **LCD_intgerToString:** Display the required decimal value on the screen
- **LCD_print_float:** Display the required floating value on the screen
- **LCD_clearScreen:** Send the clear screen command

4.2.2 Buzzer

This driver is just a simple driver as it only required to make the buzzer beep when the program code requires.

This driver have been made only to make our project on the type of the layered ones, therefore we can change the microcontroller easily.

We decided to connect the buzzer to **PIN (6)** of **PORT D** as it was simple for us to draw our PCB track without unwanted overlaps.

This driver consisted of (3) main functions which names and functionality shown in the upcoming lines:

- **Buzzer_Init:** Initialize Buzzer by adjusting the direction of its Pin as Output
- **Buzzer_On:** Turn on the Buzzer to start its beeping for specific action
- **Buzzer_Off:** Turn off the Buzzer to stop its beeping when needed

4.3 Explanation

This project is designed with concept of the layered architecture to make it portable with any kind of IDE and could be uploaded to other types of microcontrollers with simple edits.

4.3.1 H File

We started with the specification of the sensor and the button pins to **PORT D** with **PINS (2) and (3)** respectively. Since we are using it to generate interrupts to avoid using the blocking polling methods which could affect the functionality of the project and its calculations.

After that, we have produced some predefined named definitions to make our project easy to read and edit for other programmers:

- **RUNNING (1) / STOPPED (0):** values to track the water status
- **Rate_Mode (1) / Cost_Mode (0):** values to track the current active mode
- **REFRESH_RATE:** constant value to update the screen every 200 milli second to keep track of the updated values
- **COST_PER_LITER:** constant value to calculate the cost of water per liter in piasters according to the first price bracket of the Egyptian law

Then we started to declare global variables for the main parameters of our project, so it can be accessed in the main and interrupts' sections and generated as follows:

- **g_pulses:** store the number of pulses generated from the flow sensor
- **g_last_pulses:** store the last number of pulses generated from the flow sensor
- **g_mode:** store the mode of the screen
- **g_mode_changed:** store if the mode of the screen has changed
- **g_water_status:** store the status of the water

4.3.2 C File

We started with **INT0** and setup it to receive signal from the sensor to avoid the polling technique as it could cost us lack of efficiency which we do not want, the details of the interrupt was made as follows:

- **Setup:**
  - Configure the sensor pin as Input Pin
  - Trigger INT0 with the Raising Edge
  - Enable external interrupt pin INT0
- **Functionality:**
  - Increment the counter of the pulses received from the sensor

Then we moved to **INT1** and setup it to receive the button click to be able to generate a specific action according to the interrupt predefined action, the details of the interrupt was made as follows:

- **Setup:**
  - Configure the button pin as Input Pin
  - Trigger INT1 with the Raising Edge
  - Enable external interrupt pin INT1
- **Functionality:**
  - Water running phase: toggle the mode and change the flag indicting that the mode has changed
  - Water stopped phase: clears the counters of the number of pulses and the last number of pulses

After that, we have started to implement our main function and gone through some specific steps which are:

- **Initialization:**
  - Variables to store the parameters for further calculations
    - volume, cost, rate
  - Initialize drivers to be ready to call functions from
    - LCD, Buzzer
  - Enable global interrupts to be able to generate the pre-talked interrupts

- **Application Code:**
  - Present the text labels for the values to be introduced
    - Total, Cost, Rate
  - Calculate the values through mathematical equation
    - volume = g_pulses / 450.00
    - cost = volume * COST_PER_LITER
    - rate = 1000.00 * (g_pulses - g_last_pulses) / 450.00 / REFRESH_RATE
  - Present the values that was calculated
    - Total, Cost, Rate
  - Start the buzzer when reaches 55 litres
  - Assign the values of the counters with its new values
    - g_last_pulses, g_pulses, g_water_status
  - Delay with the refresh rate

4.4 Code Preview

4.4.1 H File

```
/*******************************************************************************
 *
 * [File Name]: Shower_Water_Monitor.h
 *
 * [Author]: Alaa Elsayed
 *
 * [Date Created]: 21 / 12 / 2021
 *
 * [Description]: Header File for System to measure:
 *
 *                              1) Volume of Water
 *
 *                              2) Cost of the Water
 *
 *                              3) Flow Rate
 *
 *                 Using Flow Sensor ( YF-S201 ).
 *
 *******************************************************************************/


#ifndef SHOWER_WATER_MONITOR_H_
#define SHOWER_WATER_MONITOR_H_

#include "micro_config.h"
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#include "lcd.h"
#include "gpio.h"
#include "std_types.h"
#include "common_macros.h"
#include "buzzer.h"

/*******************************************************************************
*                                 Definitions                                  *
*******************************************************************************/

/* Button HW Port and Pin Ids */
#define Button_PORT_ID                  PORTD_ID
#define Button_PIN_ID                   PIN3_ID

/* Sensor HW Port and Pin Ids */
#define Sensor_PORT_ID                  PORTD_ID
#define Sensor_PIN_ID                   PIN2_ID

/* Predefined values to track the water status */
#define RUNNING         1
#define STOPPED         0
```

```
/* Predefined values to track the current active mode */
#define Rate_Mode     1
#define Cost_Mode       0

/* Predefined constant values to be further used */
#define REFRESH_RATE          200.0
#define COST_PER_LITER        0.066

/*****************************************************************************
*                              Global Variable                               *
*****************************************************************************/

/* Global Variable to store the number of pulses generated from the flow
sensor */
volatile uint32 g_pulses = 0;
/* Global Variable to store the last number of pulses generated from the flow
sensor */
volatile uint32 g_last_pulses = 0;

/* Global Variable to store the mode of the screen */
volatile uint8 g_mode = 0;
/* Global Variable to store if the mode of the screen has changed */
volatile uint8 g_mode_changed = 0;

/* Global Variable to store the status of the water */
uint8 g_water_status;


#endif /* SHOWER_WATER_MONITOR_H_ */
```

4.4.2 C File

```
/*******************************************************************************
 *
 * [File Name]: Shower_Water_Monitor.c
 *
 * [Author]: Alaa Elsayed
 *
 * [Date Created]: 21 / 12 / 2021
 *
 * [Description]: Source File for System to measure:
 *
 *                              1) Volume of Water
 *
 *                              2) Cost of the Water
 *
 *                              3) Flow Rate
 *
 *                  Using Flow Sensor ( YF-S201 ).
 *
 *******************************************************************************/


#include "Shower_Water_Monitor.h"


/* External INT0 Interrupt Service Routine */
ISR(INT0_vect)
{
      /* Increment the counter of the pulses */
      g_pulses++;
} /* End ISR */


/* External INT0 Enable and Configuration Function */
void INT0_Init_Sensor(void)
{
      /* Configure INT0/PD2 as Input Pin */
      GPIO_setupPinDirection(Sensor_PORT_ID, Sensor_PIN_ID, PIN_INPUT);

      /* Trigger INT0 with the Raising Edge */
      MCUCR |= ( 1 << ISC00 );
      MCUCR |= ( 1 << ISC01 );

      /* Enable external interrupt pin INT0 */
      GICR  |= ( 1 << INT0 );
} /* End INT0_Init */


/* External INT1 Interrupt Service Routine */
ISR (INT1_vect)
{
      /* If the water is running, the button should toggle the mode */
```

```
	if ( g_water_status == RUNNING )
	{
		g_mode = (g_mode) ^ (1); /* Toggle the current mode */
		/* Raise a flag indicate that the mode has changed */
		g_mode_changed = 1;
	}
	/* If the water is stopped, the button should clear the counters */
	else
	{
		g_pulses = 0;		/* Clear the total number of pulses */
		g_last_pulses = 0;	/* Clear the last total number of pulses */
	}
} /* End ISR */


/* External INT1 Enable and Configuration function */
void INT1_Init_Button(void)
{
	/* Configure INT1/PD3 as Input Pin */
	GPIO_setupPinDirection(Button_PORT_ID, Button_PIN_ID, PIN_INPUT);

	/* Trigger INT1 with the Raising Edge */
	MCUCR |= ( 1 << ISC10 );
	MCUCR |= ( 1 << ISC11 );

	/* Enable external interrupt pin INT1 */
	GICR  |= ( 1 << INT1 );
} /* End INT1_Init */



/* function main begins program execution */
int main(void)
{
	/********** Initialization Code **********/

	float32 volume;	/* Variable to store the total volume of the water */
	float32 cost;	/* Variable to store the total cost of the water */
	float32 rate;	/* Variable to store the total rate of the water */

	Buzzer_Init();	/* Initialize the Buzzer */
	LCD_init();	/* Initialize the LCD */


	SREG |= ( 1 << 7 ); /* Enable interrupts by setting I-Bit */

	INT0_Init_Sensor(); /* Enable Sensor Interrupt */
	INT1_Init_Button(); /* Enable Button Interrupt */

	/* Represent the total volume of the water */
	LCD_displayString("Total:       L");

	/* Super Loop */
	for(;;)
	{
```

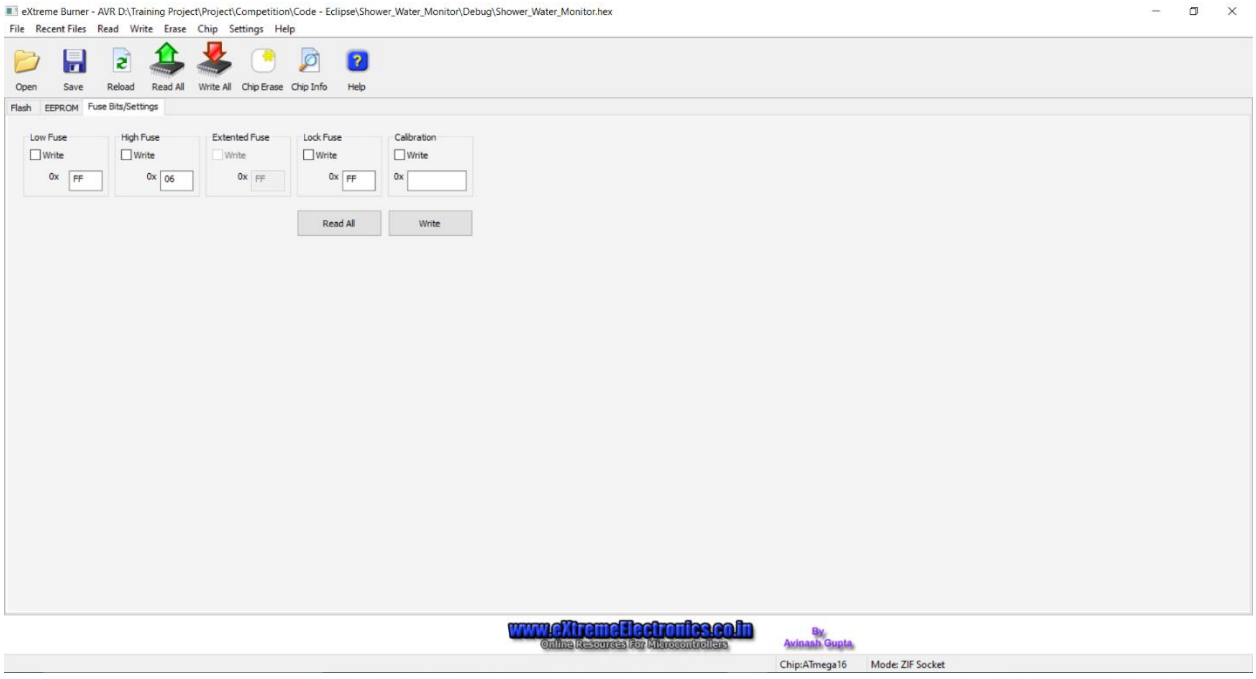
```
/********** Application Code **********/

/* If the numbers of the screen became zero avoid overwrite */
if(g_pulses == 0 && g_last_pulses == 0)
{
	LCD_moveCursor(0, 10);
	LCD_displayString("  ");
}



/* If the mode has changed, Update the text label */
do
{
	/* Move the cursor to the second line */
	LCD_moveCursor(1, 0);

	/* If the new mode is on rate mode */
	if ( g_mode == Rate_Mode )
	{
		/* Represent the flow rate text label */
		LCD_displayString("Rate:        L/s");
	}
	else /* If the new mode is on cost mode */
	{
		/* Represent the cost text label */
		LCD_displayString("Cost:        Pt.");
	}

	/* Reset the flag indicating that the mode has changed */
	g_mode_changed = 0;

} while( g_mode_changed == 1 );

/* Calculate the total volume of the water */
volume = g_pulses / 450.00;

/* Move the cursor to the first line to represent volume */
LCD_moveCursor( 0, 7 );
/* Show volume in its proper location */
LCD_FloatToString(volume);

/* If the Person exceeds the average amount of water during
shower ( 20 Gallons ) */
if (volume >= 55.00 )
{
	/* Turn ON the buzzer to alarm him to stop */
	Buzzer_On();
}
else
{
	/* If not Keep the buzzer OFF */
	Buzzer_Off();
}
```

```
            /* Move the cursor to represent the other  value */
            LCD_moveCursor( 1, 7 );

            if ( g_mode == Rate_Mode ) /* If the new mode is on rate mode */
            {
                  /* Calculate the flow rate of running water */
                  rate = 1000.00 * ( g_pulses - g_last_pulses ) / 450.00 /
                  REFRESH_RATE;
                  /* Show flow rate in its proper location */
                  LCD_FloatToString(rate);
            }
            else /* If the new mode is on cost mode */
            {
                  /* Calculate the total cost of the water */
                  cost = volume * COST_PER_LITER;
                  /* Show cost of the water in its proper location */
                  LCD_FloatToString(cost);
            }

            /*
             * Compare last and current total number of pulses
             *          -> if they are the same then the water is stopped
             *          -> if there is change then the water is running
             */
            if( g_last_pulses == g_pulses )
            {
                  g_water_status = STOPPED;
            }
            else
            {
                  g_water_status = RUNNING;
            }

            /* Set the last total number of pulses to the current number of
            pulses */
            g_last_pulses = g_pulses;

            _delay_ms(REFRESH_RATE);  /* Delay 200 ms before each update */
      } /* End Super loop */

} /* End Main Function */
```

4.5 PROGRAMMING PHASE

We have used in-circuit programming plugin as designed from the first in PCB design with 8-Mhz frequency oscillator, so we used USBasp from Atmel.

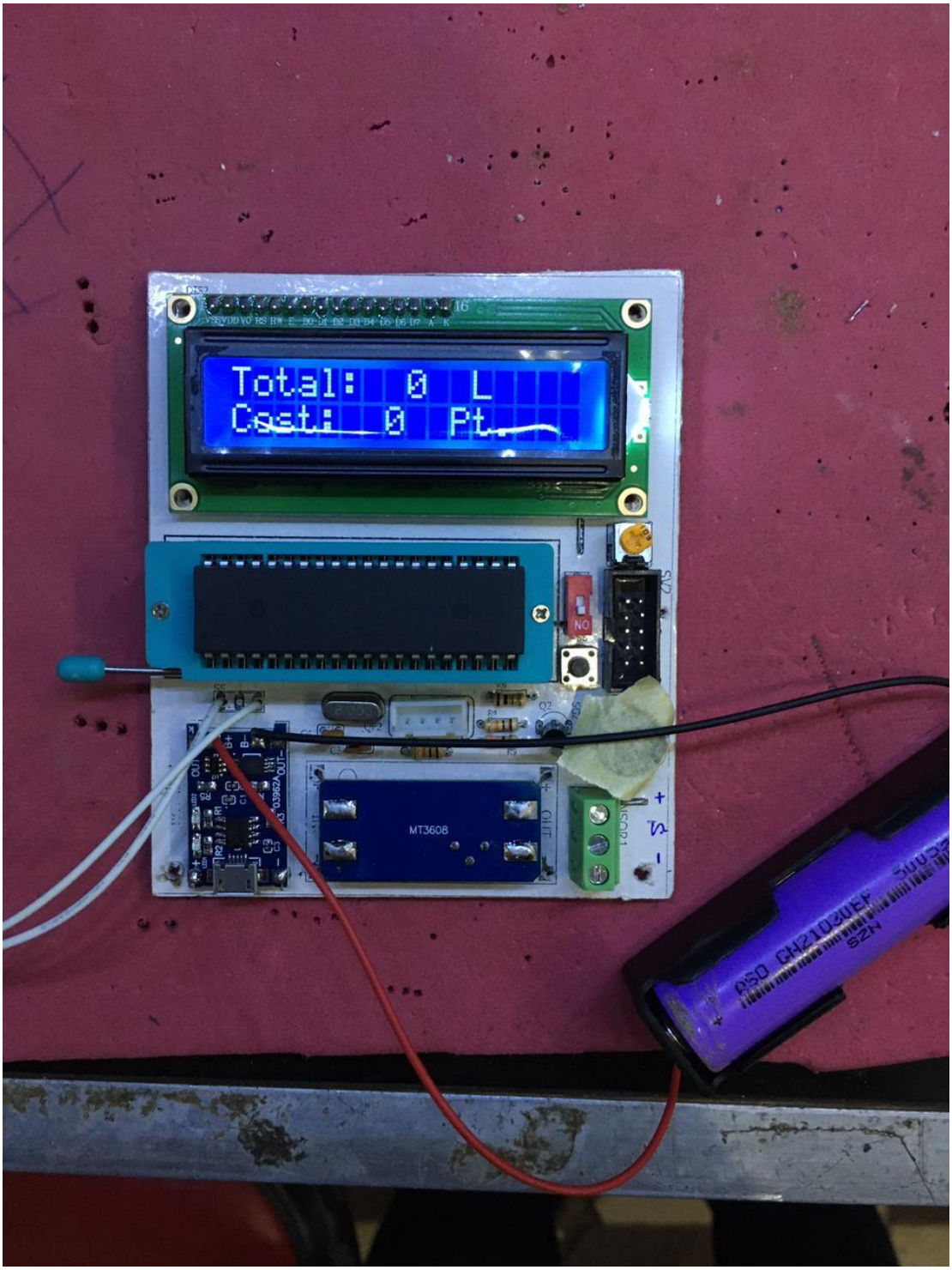
And we uploaded the project with the help of (eXtreme Burner – AVR) with following fuse bits to make our controller re-programmable:



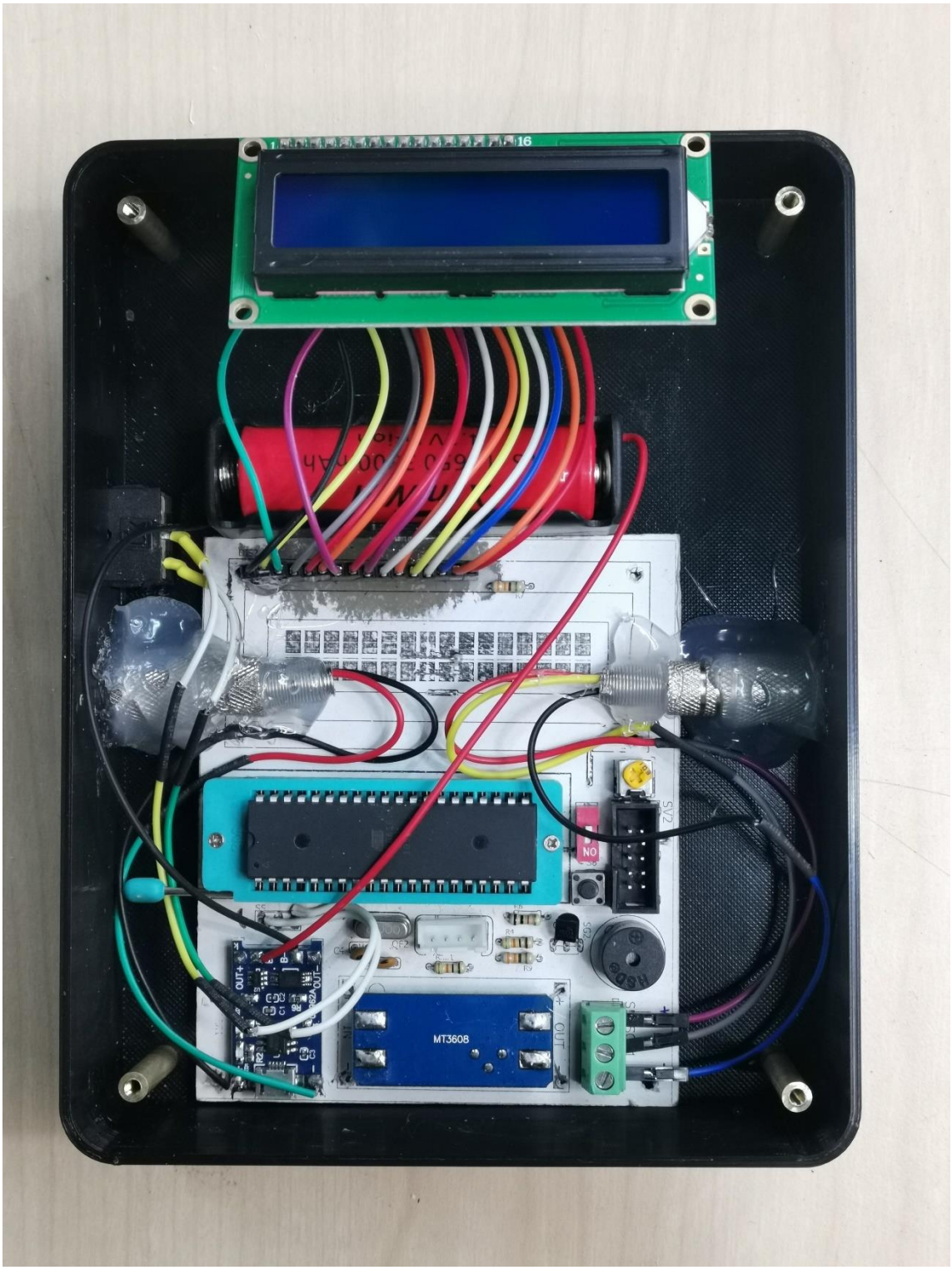
Chapter (5):

Final Product

5.1 ROAD TO FINAL PRODUCT



5.2 FINAL PRODUCT ASSEMBLY



5.3 FINAL WORKING PHASE

References

- C HOW TO PROGRAM - HARVEY DEITEL AND PAUL DEITEL
- AVR MICROCONTROLLER AND EMBEDDED SYSTEMS: USING ASSEMBLY AND, 1ST EDITION
- ATMEGA16 DATASHEET - ATMEL CORPORATION
- LCD 16X2 DATASHEET
- WATER FLOW SENSOR YF-S201 DATASHEET
- C PROGRAMMING FOR EMBEDDED SYSTEMS - KIRK ZURELL
- THE ART OF ELECTRONICS: THE X CHAPTERS - PAUL HOROWITZ AND WINFIELD HILL
- CLEAN CODE - ROBERT CECIL MARTIN
- FIRST STEPS WITH EMBEDDED SYSTEMS - BYTE CRAFT LIMITED
- AN EMBEDDED SOFTWARE PRIMER - DAVID E. SIMON