OFFICIAL USE ONLY:

PASS/FAIL_____REVIEWED BY CORDINATOR_____

DATE REVIEWED AND SIGNED _____

Name: ERIC MUINDE MULWA Reg No: E021-01-0694/2019 .

DEDAN KIMATHI UNIVERSITY OF TECHNOLOGY

STUDENTS' EXTERNAL ATTACHMENT REPORT

Promoting a culture of innovation

CENTER FOR DEVELOPMENT OF ELECTRONIC DEVICES (CDED)

DEDAN KIMATHI UNIVERSITY OF TECHNOLOGY,

PRIVATE BAG – 10143, DEDAN KIMATHI NYERI

Cded@dkut.ac.ke +254722956166

DATE: 13TH FEB – 8TH APR 2023

S T U D E N T ‘S P A R T I C U L A R S

Name of student: **MULWA ERIC MUINDE** .

Registration No. of the student: **E021-01-0694/2019** ,

Department: … ELECTRICAL AND ELECTRONICS ENGINEERING ,

Course of study: … BSC ELECTRICAL AND ELECTRONICS ENGINEERING .

Year of study: YEAR FOUR .

Company Attached: CENTER FOR DEVELOPMENT OF ELECTRONIC DEVICES (CDED).

Station Attached: DEDAN KIMATHI UNIVERSITY OF TECHNOLOGY .

Company Address: PRIVATE BAG – 10143, DEDAN KIMATHI NYERI .

Directions to the Attachment: DEDAN KIMATHI UNIVERSITY OF TECHNOLOGY .

MAIN CAMPUS, NYERI .

Name of Company Supervisor: MR. JULIUS M. KARANJA .

Cell Phone of Company Supervisor: 0722956166 .

Duration: From: 13TH FEB 2023 To: 8TH APR 2023 .

{ }

{ }

Official rubber stamp: { }

{ }

An Industrial attachment training report submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Electrical and Electronics Engineering, Dedan Kimathi University of Technology.

DECLARATION

I, ERIC MUINDE MULWA, do hereby declare to the best of my knowledge that this Project Report is my original work done at Center for Development of Electronic Devices, (CDED), Dedan Kimathi University of Technology, Nyeri. This report is prepared with no other than the indicated sources and has not been presented in any University or institution for a degree or any other reward. Moreover, to the best of my knowledge and belief, it contains no material previously published or written by another person, except where due reference is made.

SIGNED: ..... **DATE:** .....

NAME: ERIC MUINDE MULWA **REG NO:** E021-01-0694/2019

SUPERVISOR

SIGNED: ..... **DATE:** .....

NAME: MR. JULIUS M. KARANJA

ACKNOWLEDGEMENT

First of all, I would like to thank Almighty God for His blessings and for granting me charitable time, strength and protection to magnificently complete my second external attachment at Center for Development of Electronic Devices (CDED). I could have done nothing if it wasn't for His will power. I would also like to express my heart-felt gratitude to:

- My family for their moral and financial support every time I needed it throughout the attachment period. All the tokens of encouragement kept me focused and may God`s blessings be upon you; I love you all.
- My supervisor and head of CDED Center, Mr. Karanja, for giving me the opportunity to be part of the most productive team I have ever worked with in the field of Engineering and Technology. Thank you, Sir.
- All the fellow students from different Engineering Departments in the university attached at the center for their help whenever I needed it.
- My fellow Project Group Members: Esther Mukite, Irene Korir, Angela Waithera & Daniel Nyakundi. I greatly thank you for the team work and words of encouragement. In deed we learned a lot together.

It is also my pleasure to recognize and thank Dr. Waweru for recommending me to the Center for Development of Electronic Devices (CDED). As you promised me in the beginning of the attachment, I indeed learned a lot of things. I am very grateful Dr.

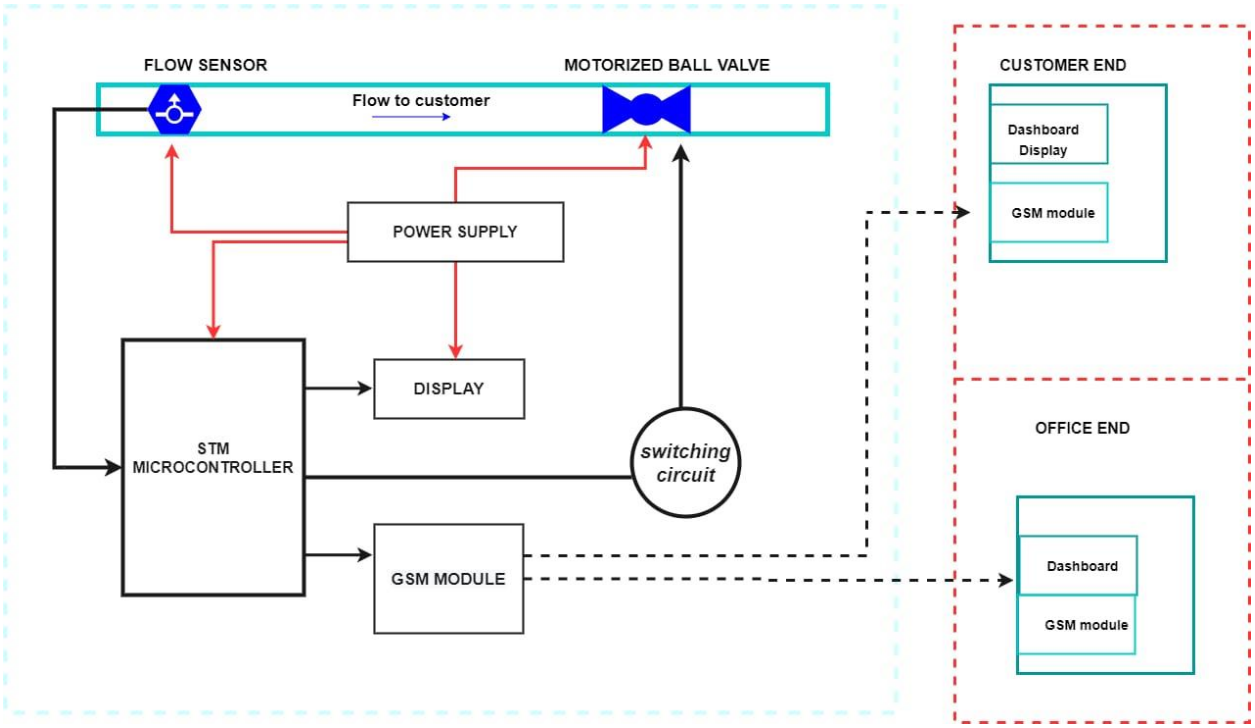
More appreciations go to my Institution supervisor Mr. Isaac Warutumo for his guidance and assessment during the attachment period. I look forward to more growth encounters with you Sir.

DEDICATION

I wish to dedicate this work to my lovely mum, Jacinta, a very special woman in my life. Mom, I dedicate this report to you for showing unwavering determination and commitment to see me succeed, even when it seemed impossible. Your tireless efforts and belief in me have always been my motivation to continue thriving in the world of Engineering and Technology. Mom, your love and sacrifice have made all the difference and I dedicate this report to you as a tribute to your unwavering support, guidance and love. Thank you.

TABLE OF CONTENTS

PROJECT BLOCK DIAGRAM & GANTT CHART



By: Eric Mulwa

Project Start Date 13-Feb-23 **GANTT CHART** Week starting

Project Name: *SMART-WATER-METER PROJECT*

#	Activity	Assigned To	Start	End	Days	Status	% Done
1	General Reading Work	Group Work	13-Feb-23	20-Feb-23	6	Complete	100%
2	Learning to Use KiCad	Group Work	14-Feb-23	27-Feb-23	10	Complete	100%
3	Learning to use Stm32CubeIDE	Group Work	27-Feb-23	13-Mar-23	11	Complete	100%
4	Interfacing Stm32 with Proteus	Group Work	27-Feb-23	20-Mar-23	16	Complete	100%
5	Components Datasheet Analysis	Group Work	14-Feb-23	10-Mar-23	19	Complete	100%
6	GSM 7600CE + STM32 Test	Group Work	2-Mar-23	10-Mar-23	7	Complete	100%
7	Stm32 + 8-Dig-7Seg Display Simulation + Test	Group Work	27-Feb-23	14-Mar-23	12	Complete	100%
8	STM32 + IR Proximity Simulation + Test	Group Work	27-Feb-23	15-Mar-23	13	Complete	100%
9	Buck - Boost Coverter Simulation + Test	Group Work	21-Feb-23	10-Mar-23	14	Complete	100%
10	Stm32 + Water Flow Sesor Code & Test	Group Work	13-Mar-23	20-Mar-23	6	Complete	100%
11	Stm32 + Motorized Valve Code & Test	Group Work	13-Mar-23	20-Mar-23	6	Complete	100%
12	Stm32 + EEPROM Code & Test	Group Work	4-Mar-23	24-Mar-23	15	In progress	70%
13	Project Schematic Circuit Development In Kicad	Group Work	3-Mar-23	3-Apr-23	22	In progress	10%
14	Project PCB Circuit Development In Kicad	Group Work	4-Apr-23	10-Apr-23	5	Not started	0%
15	Project BOM and Gerber Files Generation for Fa	Group Work	11-Apr-23	15-Apr-23	4	Not started	0%
16	Smart-Water-Meter Project Version1 test	Group Work	17-Apr-23	21-Apr-23	5	Not started	0%
17	Smart-Water-Meter Project Version2 test	Group Work	23-Apr-23	30-May-23	27	Not started	0%

#	GROUP MEMBERS
1	Eric Mulwa E021-01-0694/2019 0796456877
2	Esther Mukite E021-01-0716/2019 0757737763
3	Daniel Nyakundi E022-01-1047/2020 0707377751
4	Angela Waithera E022-01-1047/2020 0707377751
5	Irene Korir E020-01-2969/2021 0718914966

KEY	
	Completed
	Not Started
	In Progress
	Blocked
◆	Completion date

Supervisor: Mr. Karanja

2023 — Feb: 13-Feb, 20-Feb, 27-Feb; Mar: 6-Mar, 13-Mar, 20-Mar, 27-Mar; Apr: 3-Apr, 10-Apr, 17-Apr, 24-Apr; May: 1-May, 8-May, 15-May, 22-May, 29-May

Sheet1

Ready Accessibility: Investigate

INTRODUCTION

Industrial Attachment was introduced to inspire the students with practical and technical skills, as a partial fulfillment for the award of a Certificate, Diploma or a Degree and to introduce the students into working life. The training time allows the students to link the theoretical principles learnt in higher learning institutions and the real life professional and technical application. It gives the students the practical skills and the work environment philosophy, to use their skills and principles learned in class to serve the institutions and the society in general. Due to the above reasons, Dedan Kimathi University of Technology has organized 8 to 12 weeks of training in any institution of students' choice to give them an opportunity to apply the skills and knowledge achieved during the course of study and to acquire new skills in managing relationship and carrying out the jobs assigned. As a result, I joined the Center for Development of Electronic Devices (CDED), to apply all the skills learned in the university and gain experience in the real-world applications of Electrical Engineering. This program allowed me to be trained, acquire crucial practical skills and cut a niche for myself in the field of Embedded systems design and PCB design.

PURPOSE

The industrial attachment program is a partial fulfillment of the requirements for the degree of Bachelor of Science in Electrical and Electronics Engineering of Dedan Kimathi University of Technology. This report therefore provides a summary of the activities and duties carried out in Center for Development of Electronic Devices (CDED), as well as the experience gained during the attachment period.

ABOUT CENTER FOR DEVELOPMENT OF ELECTRONIC DEVICES (CDED)

CDED was established to provide the platform where the university can collaborate with the society to innovate electronic devices that can solve the various problems identified in the society. This is achieved by creating linkages with clients and likeminded stakeholders and developing electronic devices in collaboration with relevant industries or companies. CDED has already been involved in a number of projects such as the development of the digital speed governor among others. This has enabled the Centre to develop manpower with capacity, knowledge, desire and ability to design electronic devices that are of high standard and quality.

CDED offers competent electronics development in a wide range of areas such as analogue and digital electronics as well as the required firmware. We develop electronic products that comply with specified properties. We design the printed circuit boards (PCBs), put the components on printed circuit boards, design housing with connections and/or connectors to comply with mechanical stability and test the products in our in-house labs. In addition to the functionality of the components and the device, we focus on documentation and approval requirements for electronics development, by the relevant government agencies.

Our Mission

To be the best electronics designers and manufacturers in Kenya and East Africa, and support the ever-growing need for innovative solutions in the country.

Why Choose CDED?

- Modern Design
- Innovative Team

- Affordable Prices
- Discounts for Bulk Orders
- Modern Equipment
- Local Manufacturing
- Extensive Support
- Rich Ecosystem

Our team is dedicated to produce only the best quality of electronics for our clients' needs.

MAJOR PROJECTS IN THE CENTER

DEKUT SPEED GOVERNOR

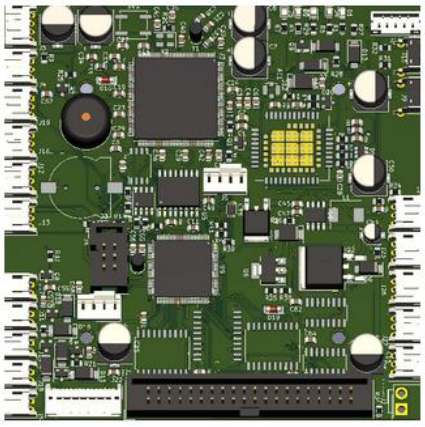
01.

- Design has been completed.
- Now awaiting approval by the relevant government bodies like KEBs and NTSA.

WATER DISPENSER CIRCUIT BOARD

02.

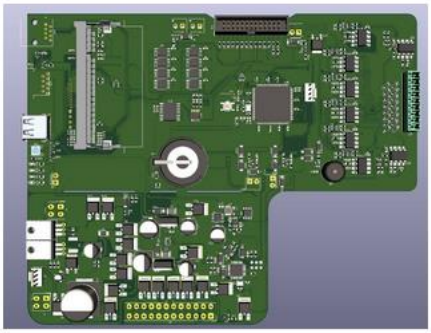
- This will be used to develop water vending machine locally which are very popular in areas with water scarcity problems.
- The system will be integrated with Mpesa so that it can operate 24/7 with minimum human operation required.



03.

VENTILATOR CIRCUIT BOARD

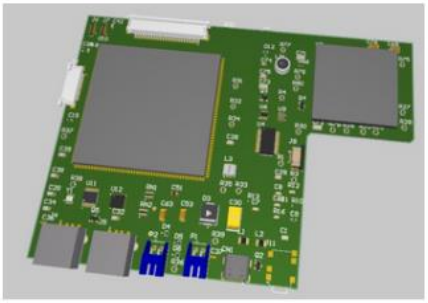
- DeKUT VENTILATOR Circuit control board already designed.
- It is awaiting production so that it can be programmed and tested.



04

STUDENT LEARNING DEVICE

- This is work in progress.
- First prototype already done.
- Currently working on some improvements. Will be ready for production by June 2021.



Contact Info

Dedan Kimathi University of Technology Private Bag – 10143, Dedan Kimathi Nyeri

cded@dkut.ac.ke

+254 722 956 166

Visit Us

We are located at Dedan Kimathi University of Technology, Nyeri.

CHAPTER 1: INTRODUCTION TO SMART WATER METER PROJECT

The first week kicked-off on 13th Feb on Monday. The head of the Center, Mr. Karanja introduced us to the Smart Water Meter Project. He generally summarized what we were required to accomplish and shortlisted some of the components to be used in the project. The Smart Water Meter Project to be designed would use water flow sensor in place of ultrasonic sensor which many meters use. The water flow sensor was availed and a motorized ball valve for opening and closing water flow. For smooth working experience in the center, we developed a working Gantt chart attached in the page 8. The project was broken down into small sections that would be completed individually. First, we had to read and familiarize ourselves with all the software and components to be used in the project. To begin with was PCB design using KiCad design suite. After a series of YouTube tutorials on PCB design and a lot of practice in KiCad, I managed to develop the following guide on PCB Design.

PCB DESIGN in KiCad: Guide by Eric Mulwa

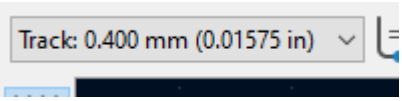
Schematic circuit and PCB design in KiCad

- File > New > Project
- Create Project folder and name it
- Open the folder and name the file
- Click on test project.seh
- Start constructing the circuit
- Click on place components and select all the components (THT- Through Hole Technolog, SMD – Surface Mounted Devices)

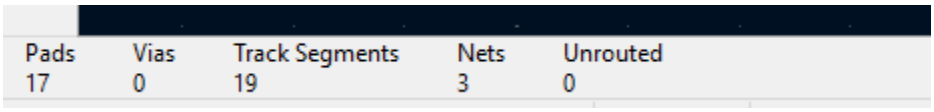
Important shortcuts include: (Click the component and select the letters)

- M for Move, G for Drag, R for Rotate, Y to mirror Vertically, X to mirror horizontally, O to auto place fields, Ctrl + E to edit symbol editor, D to show datasheet, Ctrl + X to Cut, Ctrl + C to Copy, Ctrl + V to Paste, Delete button to delete, Ctrl + D to Duplicate, Ctrl + A to select all
- Arranger the components and connect them by clicking the green straight line.
- Double click on the component values to vary them
- Click on “Run Footprint assignment tool” to assign footprints for the components.
- Click on “Apply, Save Schematic & Continue”

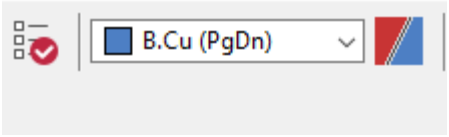
Designing the PCB

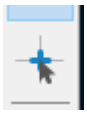
- Click “Open PCB on board editor icon”. (Similar to an actual PCB board)
- On the PCB Editor, click “Update PCB with changes made to schematic”
- Arrange the footprints according to your design outlook and make sure the tracks do not overlap each other
- Click on Track and select the width of the tracks 

- Select the front or bottom copper for routing
- Start routing the footprint tracks by clicking the “Root tracks” 
- If some tracks are still overlapping switch between front and bottom copper layers
- If all Tracks are all routed you will see a zero in the Unrouted section.



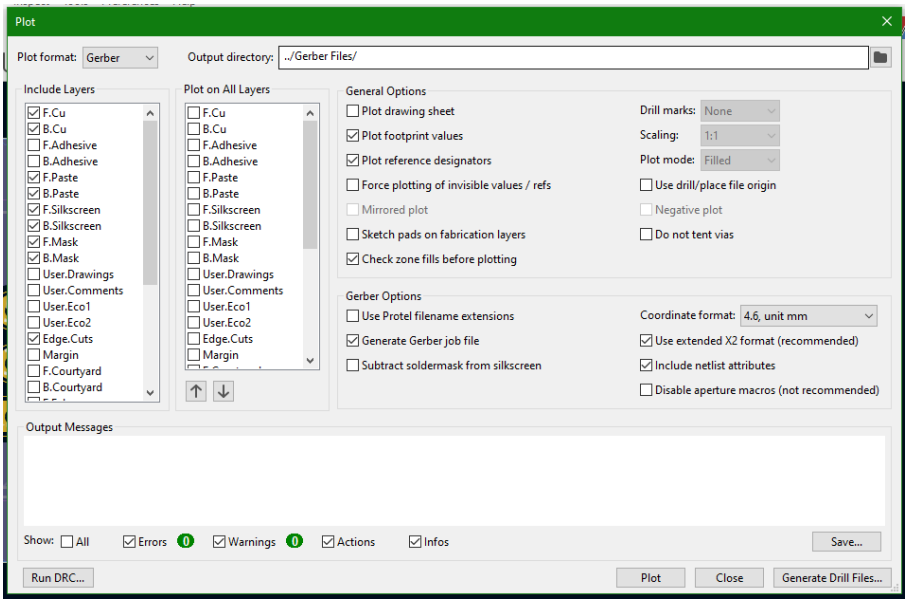
Pads	Vias	Track Segments	Nets	Unrouted
17	0	19	3	0

- Choose edge cut from this tab 

and select add traffic lines. Also select  to draw perpendicular lines.
- Select the start point of your PCB and trace a rectangle to enclose the footprints.
- You can now visualize your PCB in 3D by clicking View > 3D Viewer.
- To add PCB mounting holes, click “Add a footprint”  , and search mounting holes and place them in the respective points on the PCB.
- Edit the Blue Naming (F silk) of the Footprints and move them around to fit in the PCB board.
- To add custom names on the board, click on “Add a text item”  , then select the F silk layer and type the Name.
- The last step is adding fill zones on the board. Select the copper layer (B.CU) and click on “Add a filled zone”  and then trace the board same like for the Edge cut.
- Repeat this for the other copper layer (F.CU).
- Now your PCB board is complete.

Export the Gerber files of the board.

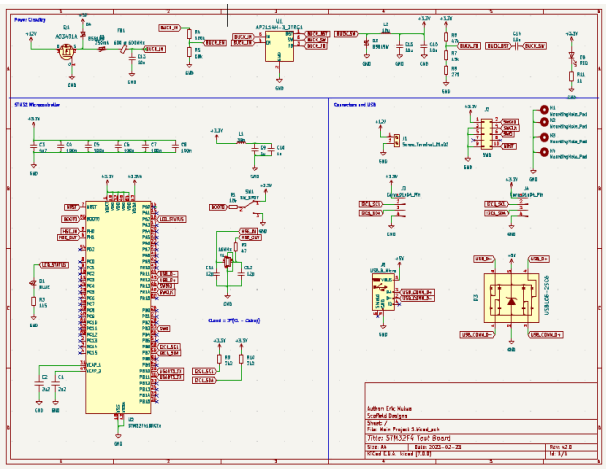
- Click on plot, 

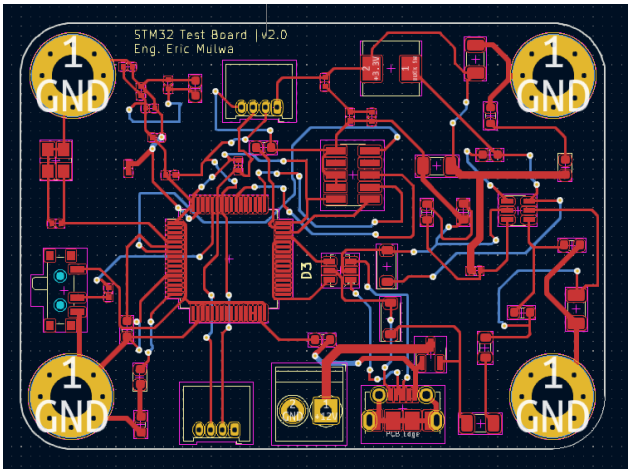
- 

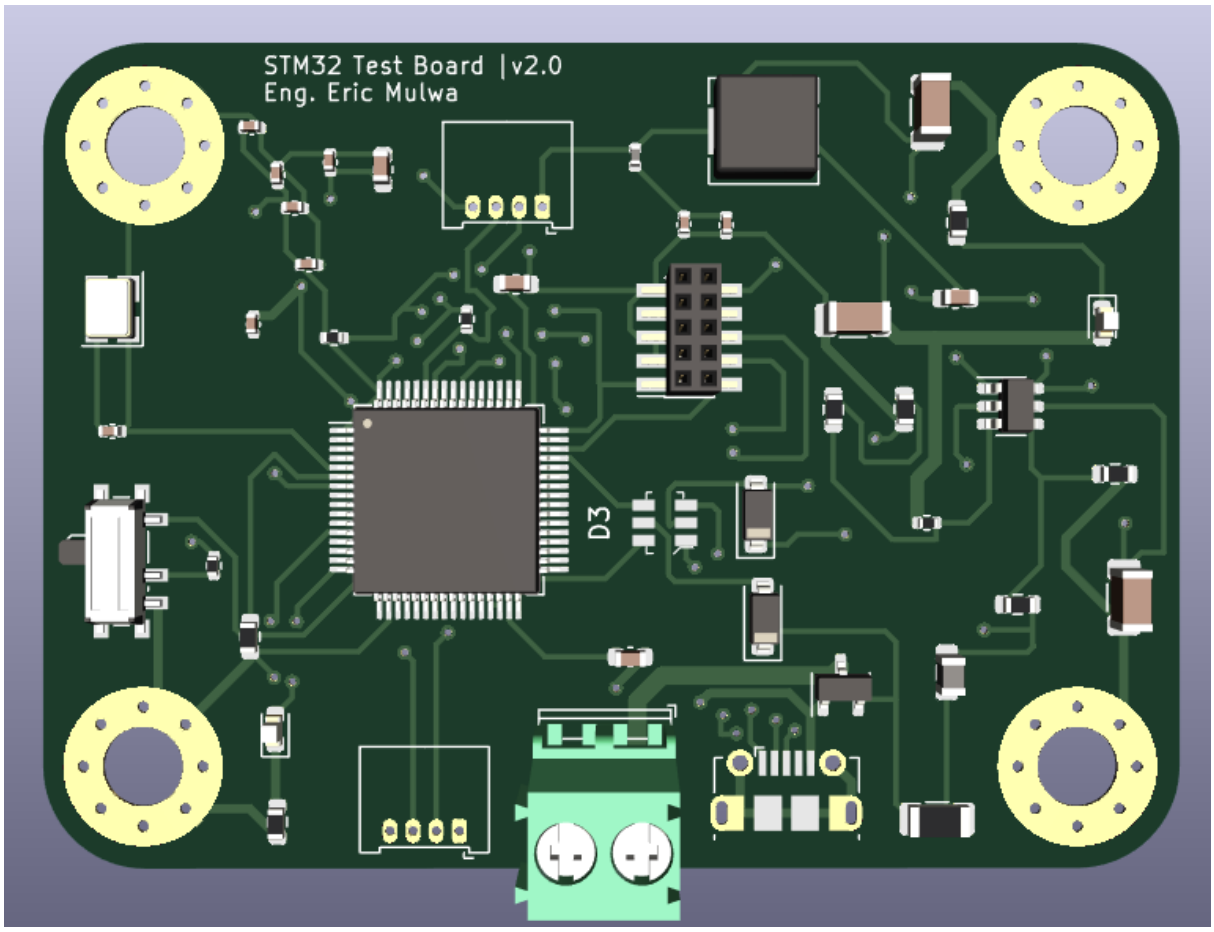
- Select the above files and click on “Plot” at the bottom, then click “Generate Drill Files” > “Generate Drill Files” > Close and Close on the Plot GUI.
- The files will be saved in the selected directory. You can take these files to a PCB printer and get your PCB.

Week One

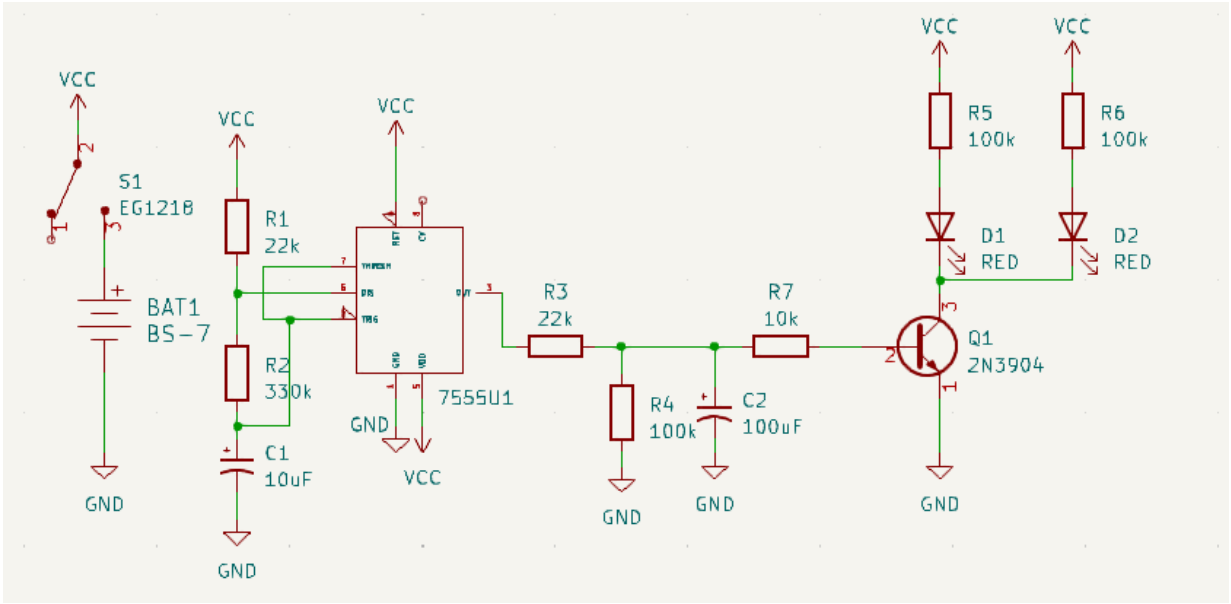
KiCad + STM32 + USB + Buck Converter







Blinking Badge Schematic Circuit



CHAPTER 2: INTRODUCTION TO STM32CUBEIDE

The second week kicked off with an introduction to STM32CubeIDE. This is an integrated development environment which allows embedded systems experts to program STM32 microcontrollers. The IDE provides an interface to develop programs to be embedded into a microcontroller in a setup for a particular function. The embedded program can then be executed by the controller and complete a particular repetitive task. The program can also be converted into a Hex file which can be used to simulate microcontrollers in some software like proteus. In the second week, we used both the two options to execute the programs developed. From the experience gained, I managed to draft a guide documenting all the important steps we followed when developing a program to be simulated in Proteus or executed in a Nucleo-board. The guide is attached below.

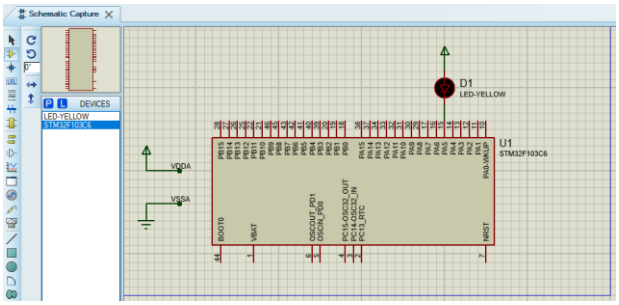
GETTING STARTED WITH STM32CubeIDE

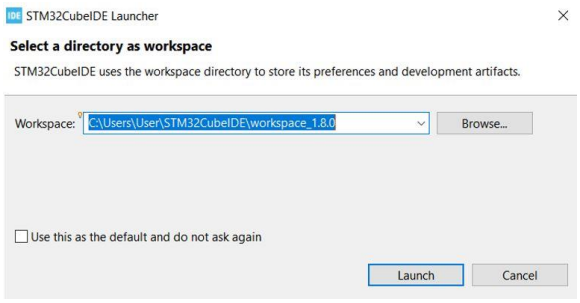
Guide by: Eric Mulwa

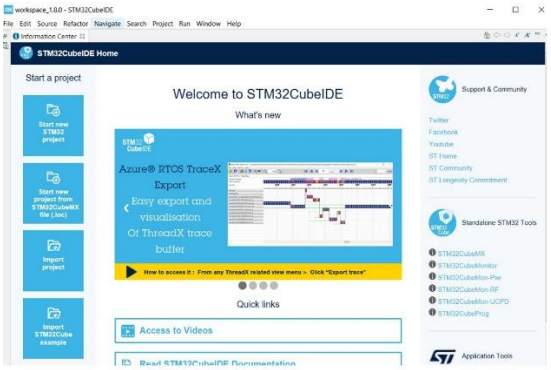
Downloading and Installing the STM32CubeIDE

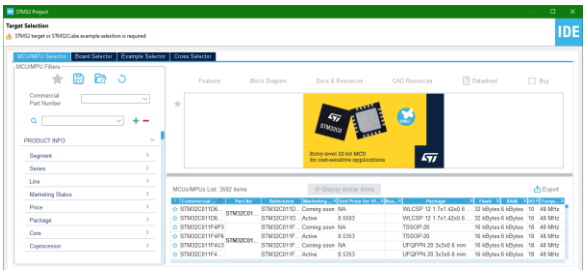
To download & Install STM32Cube IDE, go to the following link: https://www.st.com/en/development-tools/stm32cubeide.html#get-software and click 'Get latest' block according to your operating system. Run the .exe file and click okay for all the enquiries.

Programming and using the STN32F103C6 Microcontroller

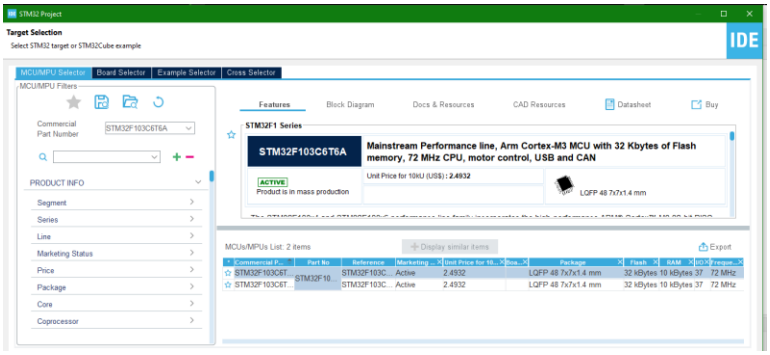
- Open proteus and select **STM32F103C6** and **LED YELLOW** from the components tab.
- On the left panel, select power and ground terminal and place them on the working space.
- Connect the circuit as shown. 

- Save the circuit and proceed to open the **STM32CubeIDE**. You will be asked to specify the directory for the workspace. Set the working directory path and set it as your default working directory.
- 

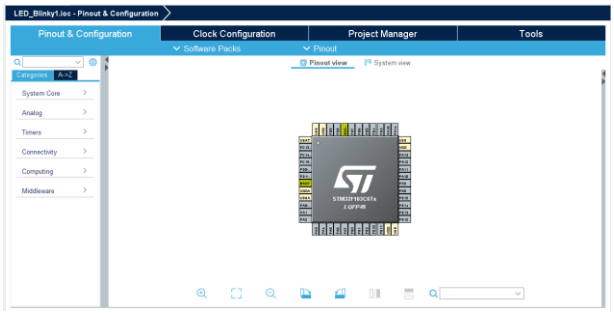
- The STM32Cube workspace will open,
- 

- Click File > New > STM32 Project and you should see the window below,
- 

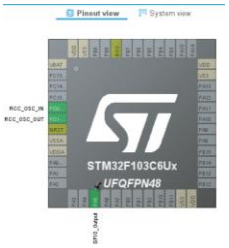
- On the "**Commercial Part Number**", Key in **STM32F103C6** and select the first item on the "**MCUs / MPUs List: 2 items**."

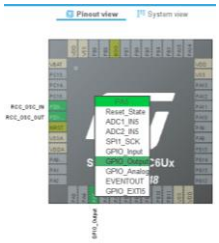
- 

- Click Next and give your Project a name, "**LED_Blinky**".
- Click Finish and allow the environment to load all the necessary libraries to configure the controller. **You will need internet connection for this step.**
- You should see the footprint below after the loading process is finished,



- This is the pin configuration for the **STM32F103C6**. The objective here is to make the LED connected on the controller in the Proteus environment to blink at a predefined interval. Select the pin **PA5** and assign it to **GPIO_Output** as shown below.





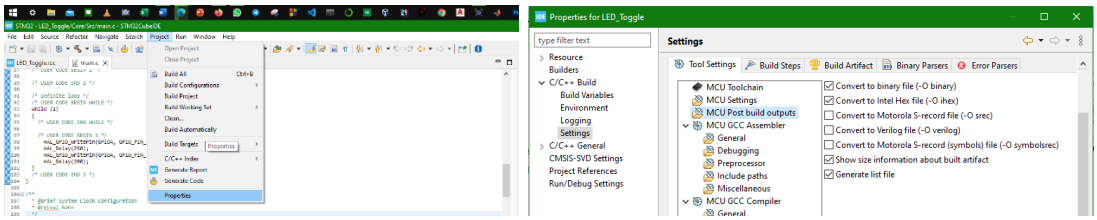
- Click **File > Save > and click yes to generate code**. You should see a **.c** program displayed with several comments.

- Move down the code to the while loop section and insert the following code just below the `/* USER CODE BEGIN 3 */`

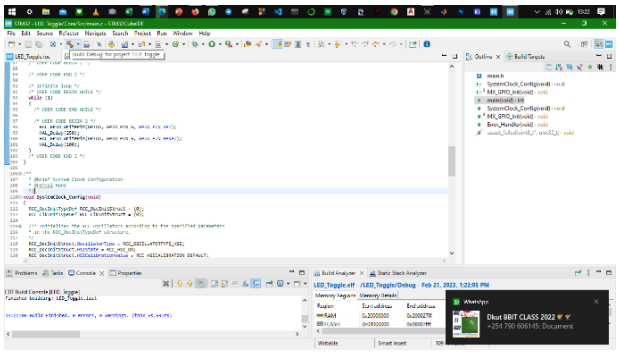
```
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
                    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
                        HAL_Delay(1000);
                        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
                        HAL_Delay(300);
  }
  /* USER CODE END 3 */
}
```

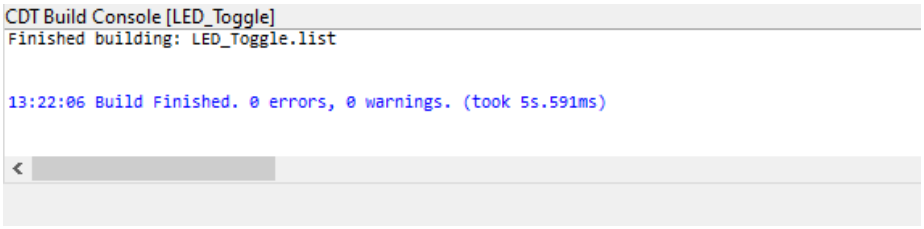
- Now what is happening is that the HAL_GPIO_WritePin() function takes in three parameters: the onboard LED port, the onboard LED pin, and the state of the pin. This function will be responsible for setting the onboard LED pin either HIGH or LOW.
- GPIO_PIN_RESET, sets the onboard LED pin to LOW whereas when we are specifying the third parameter as GPIO_PIN_SET, it sets the onboard LED pin to HIGH. The delay part delays the interval between **ON** and **OFF** states.
- Click **Project > Properties > C/C++ Build > Settings > MCU Post Build Outputs** > and select the following options to generate **.hex file** for the proteus controller.



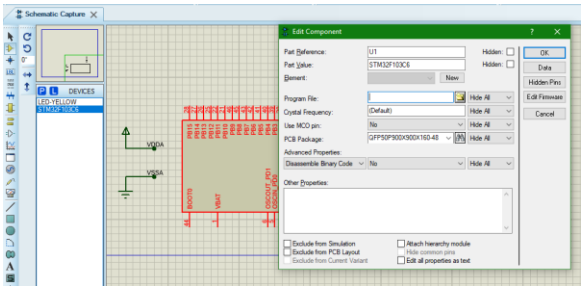
- Click "**Apply and Close** ". It will take you back to the code GUI. Click "**Debug /Build**.



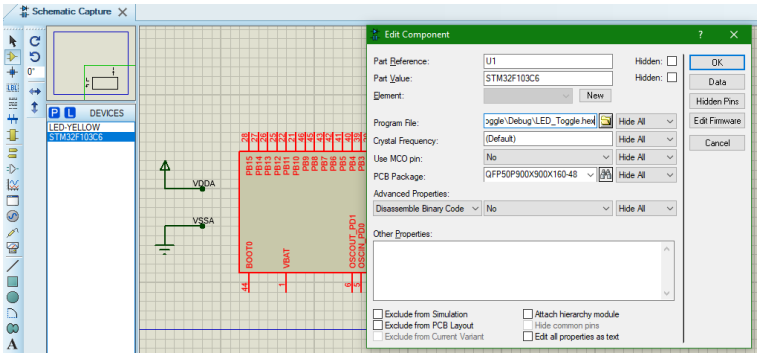
- At the bottom you should see **0 errors and 0 warnings**. This means your code is perfect and a hex file has been generated to use with the proteus **STM32F103C6.**



- Now close the IDE and go back to the proteus project. Double click on the **STM32F103C6** and you should see the GUI below pop up,

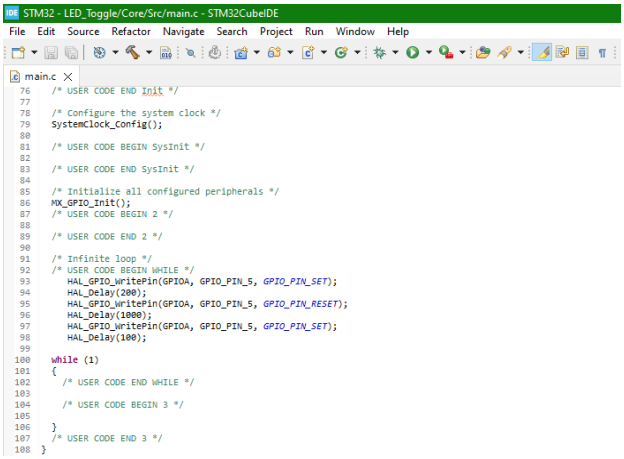


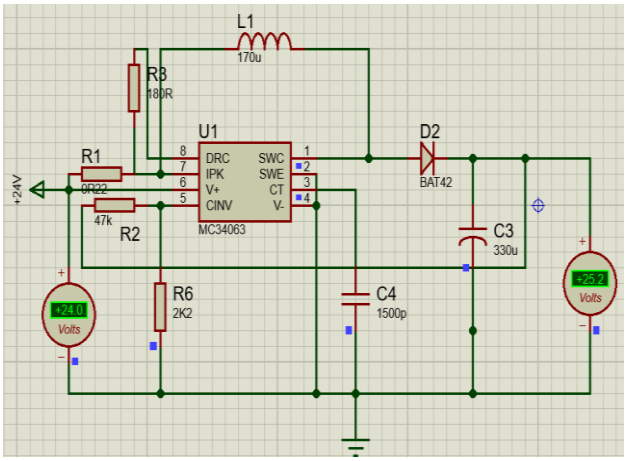
- On the "**Program File**" section, click on the little **folder icon**. Select your default **Working Directory > Project Name > Debug > LED_Blink.hex.**
- Click **Open**. You should see the GUI below with the link to the .hex file filled.

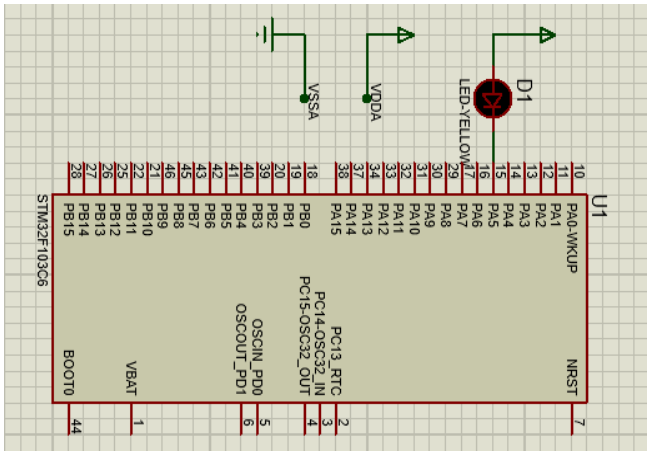


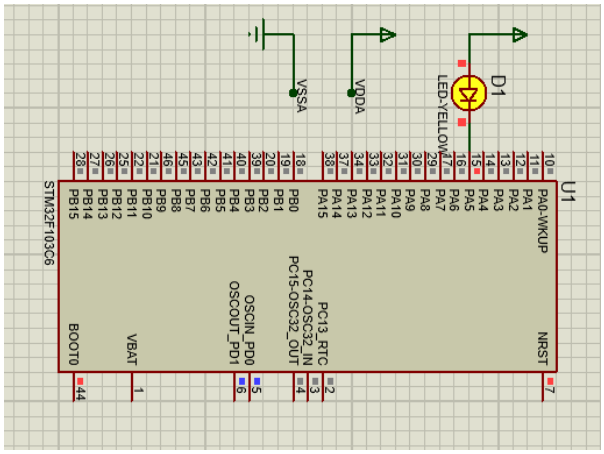
- Click "**Okay**" and run your simulation. If you followed all the steps correctly, you should see your LED blink. If it is working, **Congratulations!** Your first **STM32F103C6** project is a success. See you next time, Bye.

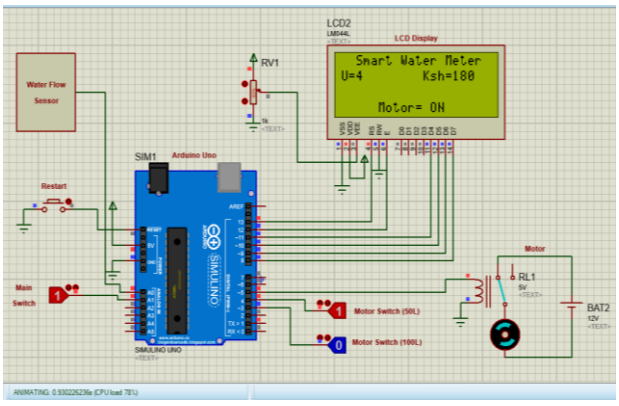
Week 2: Getting Started with STM32CubeIDE + Proteus

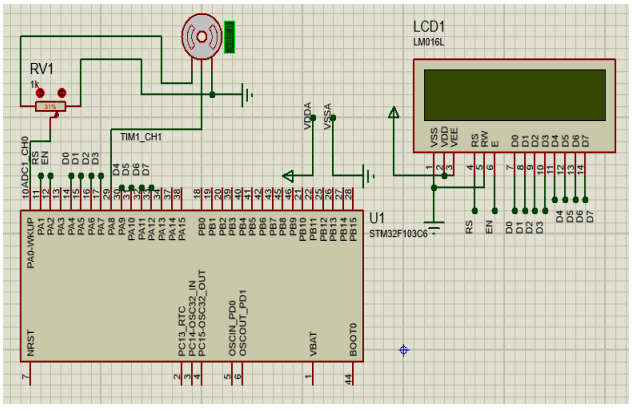












CHAPTER 3: PROJECT WORKING SCHEDULE

On the third week we embarked on the main project after intensive research and learning process on the software and the components to be used in the project. The project working schedule was broken down into two main sections; General components study and Project Implementation. The following is a project break down document I drafted to guide the group.

Working Schedule

Section 1: General Components Study

1.1 KiCad design suite – Downloading and installing.

– Basic PCB design

– Advanced PCB design.

https://www.youtube.com/watch?v=-tN14xlWWmA

1.2 STM 32 Microcontroller (STM32F411CEU6 & STM32F103C8T6)

– Architecture and configuration

– The STM32 CubeIDE Download and Installation, download link,

https://www.st.com/en/development-tools/stm32cubeide.html#overview&secondary=st-get-software

– Programming the STM32 Board. Click the link for the complete guide,

https://smartsolutions4home.com/how-to-program stm32/#:~:text=create%20a%20new%20project%20in,initialize%20all%20the%20necessary%20peripherals

1.3 RTC to wake up STM32 – Configuration and working,

https://community.st.com/s/article/how-to-configure-the-rtc-to-wake-up-the-stm32-periodically-from-low-power-modes

1.4 The GSM Module (3G, 3.4V – 4.2V)

– All about GSM Module, https://www.electronicsforu.com/resources/gsm-module

– GSM AT Commands https://www.electronicsforu.com/special/cool-stuff-misc/gsm-at-commands

– GSM System, (Unit, Customer End & Office End)

1.5 The Power Supply Unit

– Lithium ion, 3.7V Battery Unit

– Boost converter (from 3.7V to 5V)

– O.D Protection

– Battery level IC

1.6 Display (Segment LCD 2.2 – 3V) HT1621

1.7 Motorized Ball Valve (5V)

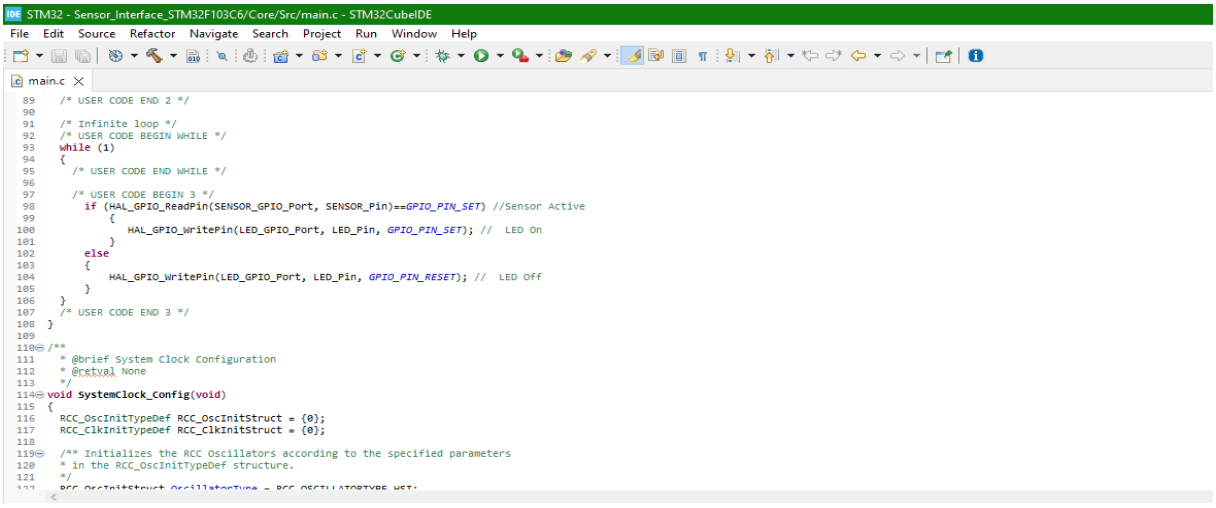
1.8 Flow Sensor (3.1V)

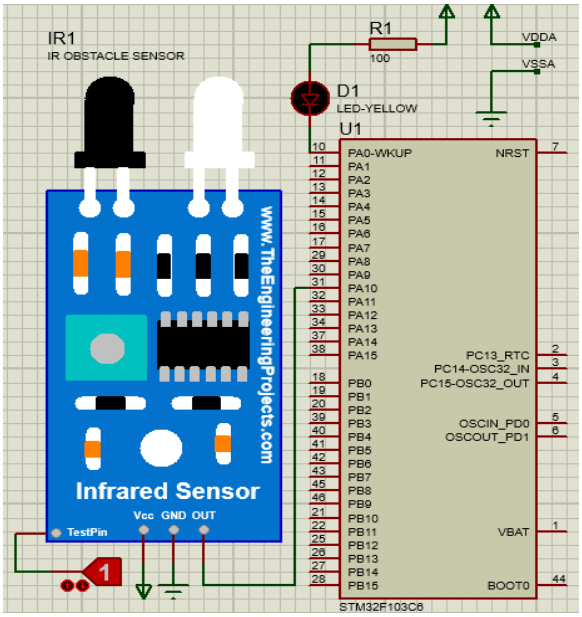
To achieve the above working schedule objectives, a Gantt Chart was drafted. It can be found in page 8 of the report with the project block diagram. In the block diagram, the STM32 is the heart of the project as it collects data from the water flow sensor and processes it. The water flow sensor works on the principle of the hall effect sensor. It has a turbine which is rotated by flowing water. In the middle of the fan, there is a permanent magnet which rotates together with the turbine. On top of the turbine is a hall effect sensor which generates voltage pulses when the magnet is rotated. Th rotating magnet field isolated electrons from the protons in the hall effect sensor plate and causes potential difference. The generated pulses are used to measure the volume of water passing through the pipe at a particular time interval. This volume has a

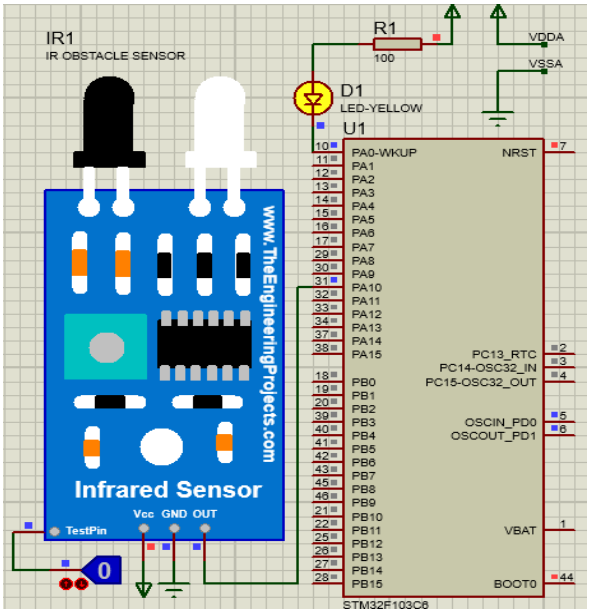
mathematical correlation with the frequency of the pulses. This mathematical computation is completed in an embedded program in the project. After computing the volume of water consumed for a particular time interval, the data stored in a EEPROM connected to the controller. The data is then fetched and displayed in HT1621 segment display and also send wirelessly using the GSM SIM7600CE module to the server and the customer end. The office end calculates cost of water consumed and shares with the consumer to make payments accordingly.

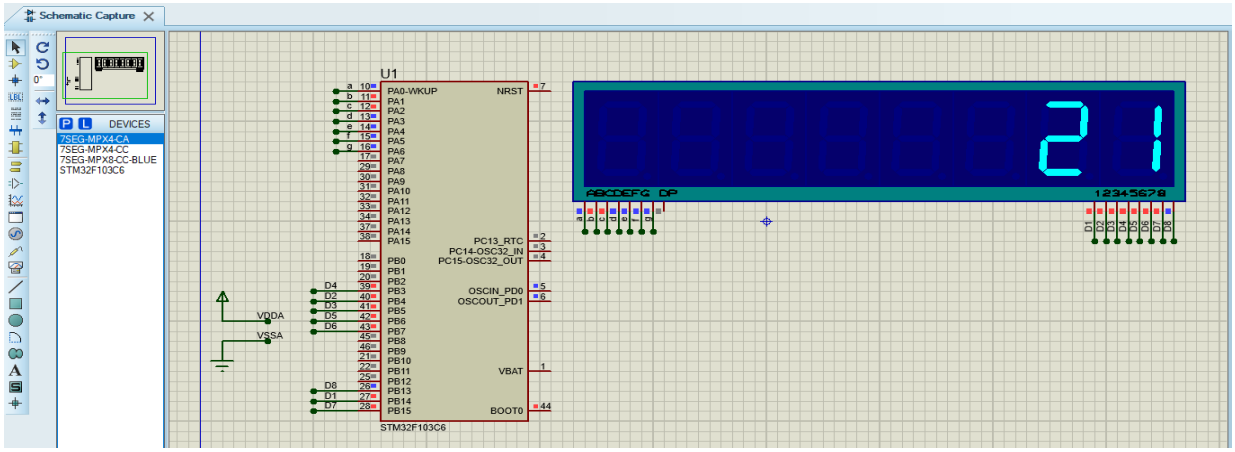
In the event the consumer defaults to pay, the office end can instruct the STM32 microcontroller to cut off the supply of water by closing the motorized ball valve. This is done using AT Commands with the GSM module and a phone. When the defaulter heeds and clears the bill, the office end sends another command to open the ball valve and allow water supply again. To achieve this working process, the project was broken down into small tasks to be completed individually. The first task was to design a Boost converter that is able to step up 3.7V to 5V. The designed circuit was simulated in proteus. The second task was to include a power LED in the main board of the project to indicate when the project is powered and for two seconds and go off. The following tasks are as shown in the Gantt Chart on page 8 of this report.

Week Three Attachments









CHAPTER 4: STM32F411CEU6 BLACK PILL

On week 4, we proceeded to developing programs in the STM32CubeIDE to test the functionality of all the project components. But first we downloaded all the components datasheets and went through them thoroughly in order to know how to develop effective and fast to execute programs and how to interface them with the STM32 microcontroller. There are several STM32 microcontrollers and we had trouble choosing which was best fit for our project. Therefore, we downloaded several data sheets of the most used STM32 microcontrollers and went through their specifications and architecture. After pondering over the program storage memory, we downloaded the STM32H7743 series data sheet. This microcontroller is 32-bit Arm Cortex based, M7 400MHz MCUs, up to 2MB Flash, 1MB RAM, 46 com and analog interfaces. This chip is very powerful and can support even ethernet ports. It has 160 pins. The datasheet was 231 pages long showing it was high tech and using it on a small embedded system would be uneconomical.

Our finals decision on the STM32 microcontroller to be used was STM32F411CEU6 Black Pill. The STM32F411CEU6 Black Pill is a high-performance access line, Arm Cortex – M4 Core with DSP and FPU, 512 Kbytes of Flash memory, 100MHz CPU and ART accelerator. The chip costs $ 3.9073/=. It also implements a full set of DSP instructions and a memory protection unit (MPU) which enhances application security. The chip has 128 Kbytes of SRAM and an extensive range of enhanced I/Os and peripherals connected to two APB buses, two AHB bus and a 32-bit multi – AHB bus matrix. After some research and reading work, we developed programs send sms using GSM+STM32F411CEU6 and GSM+Arduino UNO and executed them accordingly. The attachments below show the work done and executed.

Week Four: GSM+STM32 & GSM+ARDUINO UNO

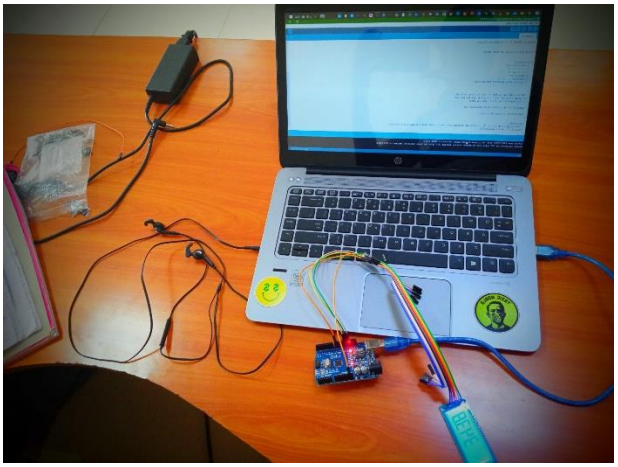
STM32 - GSM_F411_Test/Core/Src/main.c - STM32CubeIDE

File Edit Source Refactor Navigate Search Project Run Window Help

main.c main.c *main.c

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USART1_UART_Init();
/* USER CODE BEGIN 2 */
char mobileNumber[] = "+254796456877";  // Enter the Mobile Number you want to send to
char ATcommand[80];
uint8_t buffer[30] = {0};
uint8_t ATisOK = 0;
while(!ATisOK){
      sprintf(ATcommand,"AT\r\n");
      HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
      HAL_UART_Receive (&huart1, buffer, 30, 100);
      HAL_Delay(1000);
      if(strstr((char *)buffer,"OK")){
          ATisOK = 1;
      }
      HAL_Delay(1000);
      memset(buffer,0,sizeof(buffer));
}
sprintf(ATcommand,"AT+CMGF=1\r\n");
HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
HAL_UART_Receive (&huart1, buffer, 30, 100);
HAL_Delay(1000);
memset(buffer,0,sizeof(buffer));
sprintf(ATcommand,"AT+CMGS=\"%s\"\r\n",mobileNumber);
HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
HAL_Delay(100);
sprintf(ATcommand,"Hello there, GSM started,%c",0x1a);
HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
HAL_UART_Receive (&huart1, buffer, 30, 100);
memset(buffer,0,sizeof(buffer));
HAL_Delay(4000);
/* USER CODE END 2 */
```

Console



HelloWorld | Arduino 1.8.19 (Windows Store 1.8.57.0)

File Edit Sketch Tools Help

HelloWorld

```
#include <HT1621.h> // include our library


HT1621 lcd; // create an "lcd" object


void setup(){
  // start the lcd:
  // cs to pin 13
  // wr to pin 12
  // Data to pin 8
  // backlight to pin 10
  // you can chose whichever pin you want


  //lcd.begin(13, 12, 8, 10); // (cs, wr, Data, backlight)
  // if no backlight control is given, you can also use:
  lcd.begin(13, 12, 8); // (cs, wr, Data)

  //lcd.backlight(); // turn on the backlight led

  lcd.clear(); // clear the screen
}

void loop(){
  lcd.print(millis()/1000.0, 0); // print the floating point seconds value with 3 decimal digits precision
  delay(1); // wait 1 millisecond
}
```

Done compiling.

```
Sketch uses 3910 bytes (12%) of program storage space. Maximum is 32256 bytes.
Global variables use 121 bytes (5%) of dynamic memory, leaving 1927 bytes for local variables. Maximum is 2048 bytes.
```

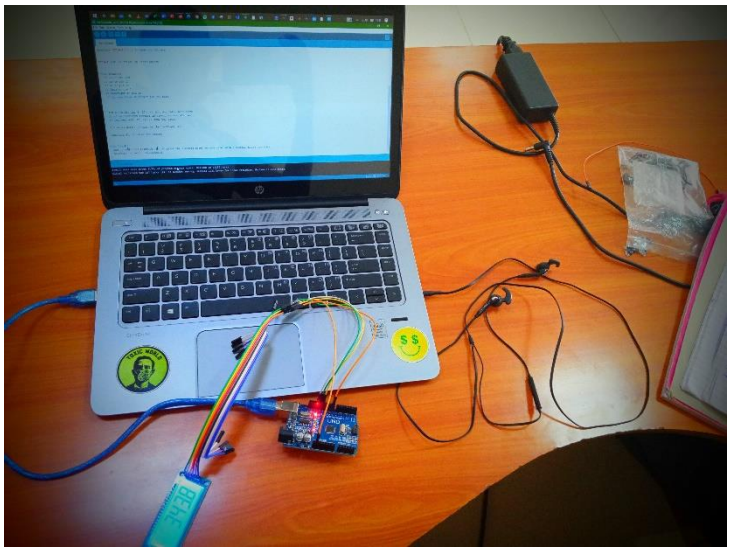
1 Arduino Uno on COM6

CHAPTER 5: HT1621 SEGMENT DISPLAY

On week 5, we tested the HT1621 Segment Display with Arduino UNO and the STM32F411CU6. After a lot of trials using different HT1621 Libraries and different programs, we were able to achieve a working set up and smoothly running programs as shown below.

Arduino + HT1621 Test

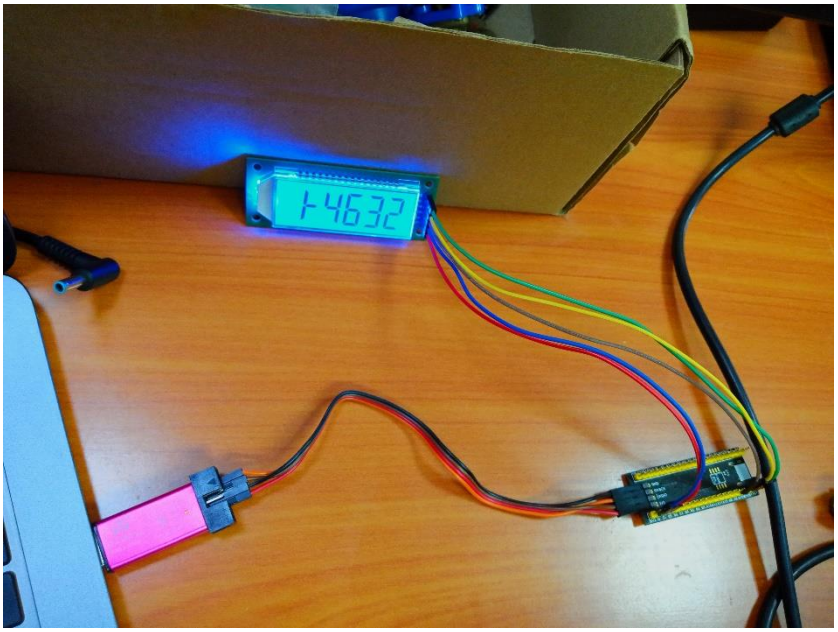
```
#include <lcdlib.h> //library for display control
lcdlib lcd; //instance of display controller
/*0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b, C, c, d, E, F, H, h, L, n, N, P, r, t, U, -, bat, pf, ,*/
const char
num[]={0x7D,0x60,0x3E,0x7A,0x63,0x5B,0x5F,0x70,0x7F,0x7B,0x77,0x4F,0x1D,0x0E,0x6E,0x1F,0x17,0x67,0x47,0x0D
,0x46,0x75 , 0x37, 0x06,0x0F,0x6D,0x02,0x80,0xFF, 0x00 };
/*index num[i] /*0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 2*/
void setup()
{
     lcd.run(2,3,4,5); //[ cs wr data led+] pin definition
     lcd.conf(); //initial setting
     lcd.clr(); // clear the display
     //write HELLO
     lcd.display(10, num[17]);
     lcd.display(8, num[15]);
     lcd.display(6, num[19]);
     lcd.display(4, num[19]);
     lcd.display(2, num[0]);
     //end HELLO
     delay(3000);
     lcd.clr(); // clear the display
}
void loop()
{
     delay(1000);
     // will write all the symbols of our array
     for(int i=0; i<5; i++) {
         lcd.display(0, num[i]);
         lcd.display(2, num[i]);
         lcd.display(4, num[i]);
         lcd.display(6, num[i]);
         lcd.display(8, num[i]);
         lcd.display(10, num[i]);
         delay(1000);
     }
     lcd.clr(); // clear the display
     //write numbers with FLOATING POINT
     lcd.dispnum(3.21); delay(1500);
     lcd.dispnum(980.15); delay(1500);
     lcd.dispnum(3214.58); delay(1500);
     lcd.dispnum(85641.2);
     delay(1500);
     lcd.clr(); // clear the display
}
// function that writes loop on the display
//writeLoop()
{
     //write ERIC
     lcd.display(10,num[15]);
     lcd.display(8, num[23]);
     lcd.display(6, num[1]);
     lcd.display(4, num[12]);
     delay(5000);
     lcd.clr(); // clear the display
```



```
}
//function that writes the battery markers
//writeBattery()
{
     //write the BATTERY symbols
     lcd.display(10, num[27]);
     delay(1000);
     lcd.display(8, num[27]);
     delay(1000);
     lcd.display(6, num[27]);
     delay(1000);
     lcd.display(10, 0x00);
     delay(500);
     lcd.display(8, 0x00);
     delay(500);
     lcd.display(6, 0x00);
     delay(500);
}
}
```

STM32F411CEU6 + HT1621 TEST

```
#include "HT1621.h"
/* USER CODE END Includes */
/* Private variables ---------------------------------------------------------*/
SPI_HandleTypeDef hspi2;
/* USER CODE BEGIN PV */
const uint8_t displ_size = 4;
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_SPI2_Init(void);
/* Private user code ---------------------------------------------------------*/
void simulate_clock(uint32_t ms) {
  uint8_t  minute = (ms /500) % 60;
  uint8_t  hour   = (ms / 5000) % 24;
  HT1621_Clock(hour, minute, ms % 2);
}
void simulate_date(uint32_t ms) {
  uint8_t  day   = (ms /500) % 31;
  uint8_t  month = (ms / 2000) % 12;
  HT1621_Date(day, month);
}
void show_letters(uint32_t data) {
        uint8_t sym = data % 56;
        HT1621_Prepare();
        for (uint8_t i = displ_size; i < 255; --i) {
                HT1621_Set_Char(i, sym + 40, false);
                ++sym;
                sym %= 56;
        }
        HT1621_Show();
}
int main(void)
{
  /* USER CODE BEGIN 1 */
        uint32_t data   = 0;
        uint32_t period = 30000;
        uint32_t till   = period;
        uint8_t mode    = 0;
  /* USER CODE END 1 */
  MX_GPIO_Init();
  MX_SPI2_Init();
  /* USER CODE BEGIN 2 */
  HT1621_Init(displ_size);
  HT1621_Clear();
  HT1621_String("HELO", 0);
  HAL_Delay(1000);
  /* USER CODE BEGIN WHILE */
```

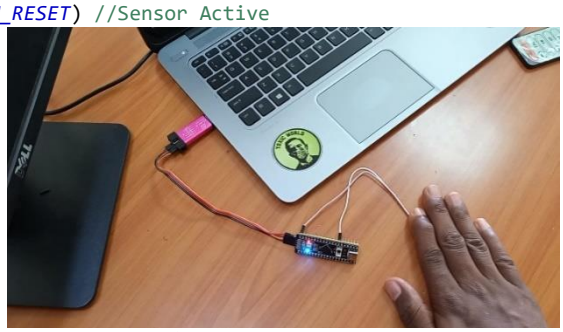


CHAPTER 6: IR PROXIMITY SENSOR AND MOTORIZED BALL VALVE

On week six we tested the working of the IR Proximity sensor with STM32F411CEU6. After achieving a working setup and prove of concept for the project, we proceeded to test the motorized ball valve. The IR sensor would be used to cut down power consumption in the final project by only displaying readings when the meter lid is opened. After testing the two components we also tested the working of the flow sensor using Arduino UNO. The programs developed for the IR sensor, the motorized ball valve and the flow sensor with Arduino are shown below with pictures of the working setups.

IR SENSOR + STM32F411CEU6 TEST

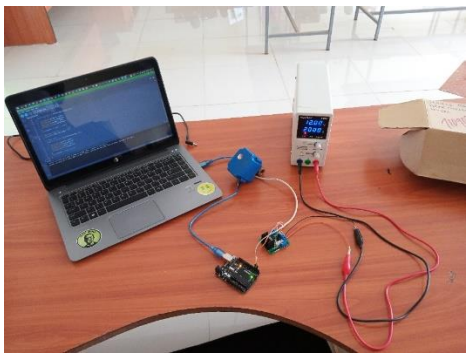
```
******************************************************************************
*/ ERIC MULWA
/* Includes
#include "main.h"
/* Private function prototypes
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
int main(void)
{
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE BEGIN 3 */
if (HAL_GPIO_ReadPin(SENSOR_GPIO_Port, SENSOR_Pin)==GPIO_PIN_RESET) //Sensor Active
{
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
  HAL_Delay(3000);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
  HAL_Delay(300);
}
          else
         {
 HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
 HAL_Delay(500);
      }
  /* USER CODE END 3 */
```



Arduino Uno + Motorized Ball Valve + Motor Driver Test

```
// Define pin connections & motor's steps per revolution
const int dirPin = 2;
const int stepPin = 3;
const int stepsPerRevolution = 500;
int stepDelay=2800;

void setup()
{
  // Declare pins as Outputs
  pinMode(stepPin, OUTPUT);
  pinMode(dirPin, OUTPUT);
}
void loop()
{
  //clockwise
  digitalWrite(dirPin, HIGH);
```



```
  // Spin motor
  for(int x = 0; x < stepsPerRevolution; x++)
  {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(stepDelay);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(stepDelay);
  }
  delay(1000); // Wait a second
}
```

STM32F411CEU6 + Motorized Ball Valve + Motor Driver Test

```
/******************************************************************************
/* ERIC MULWA
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stdio.h"
#include <string.h>
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
int main(void)
{
  while (1)
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
 //Opening Valve (Clockwise)
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
HAL_Delay(2800);
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
HAL_Delay(2000);
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);
HAL_Delay(2800);
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
HAL_Delay(2000);
  }
  /* USER CODE END 3 */
```

Arduino Uno + Water Flow Sensor Test

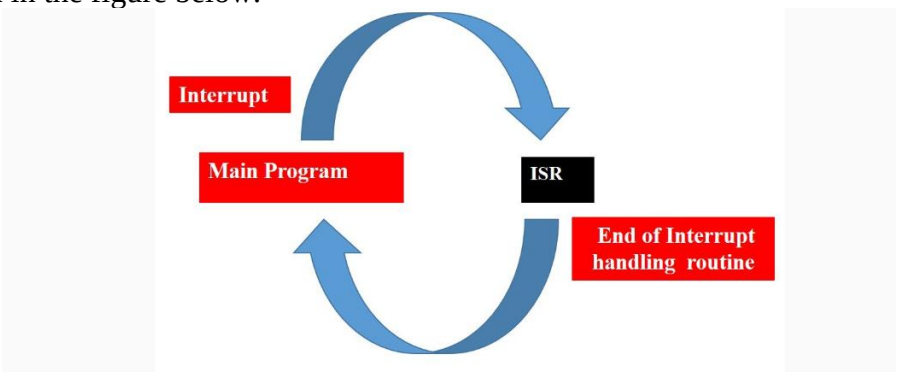
```
// include the library code:
int sensorPin = 2;
volatile long pulse;
float volume;
void setup() {
  pinMode(sensorPin, INPUT);
  Serial.begin(9600);
  attachInterrupt(digitalPinToInterrupt(sensorPin), increase, RISING);
}
void loop() {
  volume = 2.663 * pulse;
  Serial.print(volume);
  Serial.println(" mL");
  delay(500);
}
void increase() {
  pulse++;
}
```

CHAPTER 7: STM32 EXTERNAL INTERRUPT AND WATER FLOW SENSOR

After succeeding in testing the Water Flow Sensor with Arduino, we embarked on intensive research to achieve the same working principle with the STM32 microcontroller. It was never easy since there were no any references in the internet to help. We did research on how to use the External Interrupt Service Routine in STM32. Interrupts are used to handle events that do not happen during the sequential execution of a program. For example, we want to perform certain tasks and these tasks execute sequentially in your program. But there are few tasks that only execute when a special event occurs such as an external trigger signal to the digital input pin of a microcontroller.

An external interrupt or a 'hardware interrupt' is caused by the external hardware module. For example, there is a Touch Interrupt which happens when touch is detected and a GPIO interrupt when a key is pressed down. With interrupt, we do not need to continuously check the state of the digital input pin. When an interrupt occurs (a change is detected), the processor stops the execution of the main program and a function is called upon known as ISR or the Interrupt Service Routine. The processor then temporarily works on a different task (ISR) and then gets back to the main program after the handling routine has ended.

This is shown in the figure below.



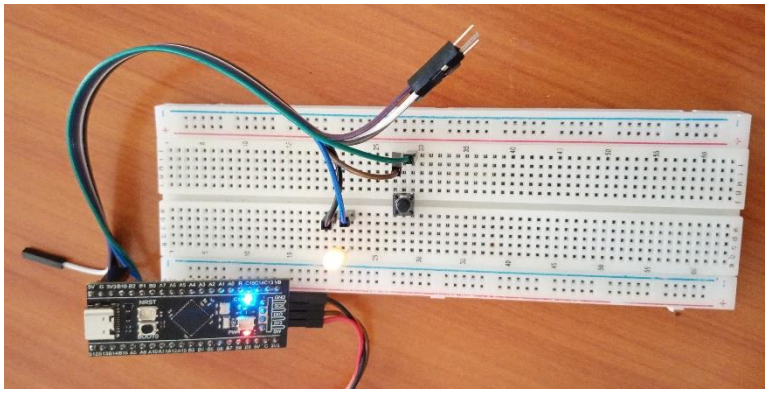
The STM32 ARM microcontroller interrupts are generated in the following manner: The system runs the ISR and then goes back to the main program. The NVIC and EXTI are configured. The

Interrupt Service Routine (ISR) also known as the interrupt service routine handler is defined to enable the external interrupts. We applied the same concept to develop a program to read voltage pulses from the water flow sensor. But first we blinked an LED using the ISR before using it for the Water Flow Sensor. The following is the programs written and the setups.

Interrupt Routine Service + LED + Push Button

```
******************************************************************************
/ ERIC MULWA
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* Private function prototypes
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
int main(void)
{
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```



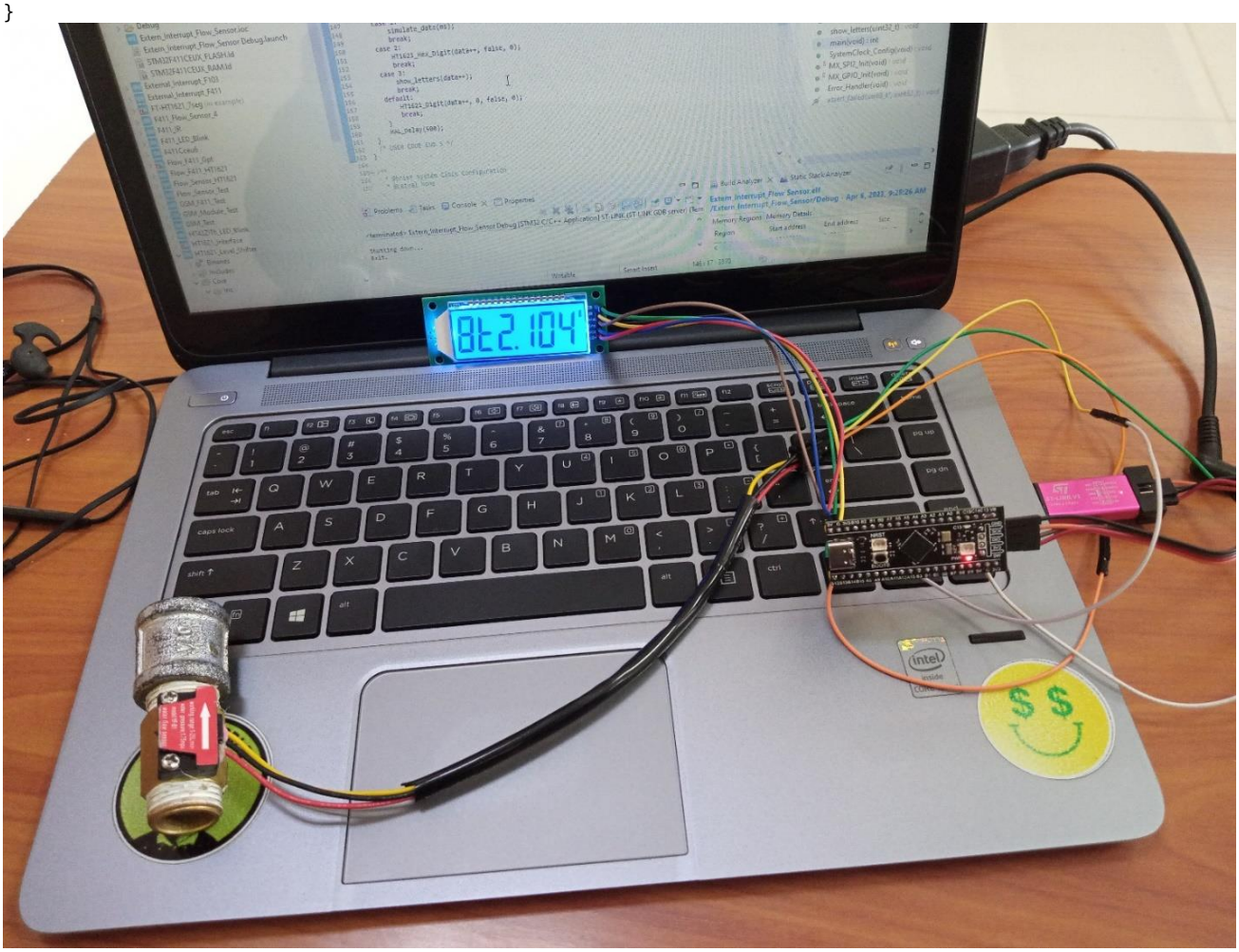
```
// External Interrupt ISR Handler CallBackFunction
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_9) // INT Source is pin A9
    {
    HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_11); // Toggle LED
    }
}
```

IRS + HT1621+ Water Flow Sensor

```
******************************************************************************
/* ERIC MULWA
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "stdio.h"
#include <string.h>
#include "HT1621.h"
/* Private variables ---------------------------------------------------------*/
SPI_HandleTypeDef hspi2;
/* USER CODE BEGIN PV */
const uint8_t displ_size = 4;
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_SPI2_Init(void);
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
volatile uint32_t pulseCount = 0; //Read interrupt pulses from the Pin

int main(void)
{
  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
```

```
  HAL_Init();
  /* Configure the system clock */
  SystemClock_Config();
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_SPI2_Init();
  /* USER CODE BEGIN 2 */
  HT1621_Init(displ_size);
  HT1621_Clear();
  HT1621_String("HELO", 9);
  HAL_Delay(1000);
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
//Display the water volume in M^3 consumed by the user
 uint32_t Volume = pulseCount * 0.003165;
                  HT1621_Digit(Volume, 0, false, 6);
              HAL_Delay(300);
  }
  /* USER CODE END 3 */
}
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) //interrupt pin incrimenting pulse count
{
    if(GPIO_Pin == GPIO_PIN_9) // INT Source is pin A9
    {
    pulseCount++;
    }
}
```



CHAPTER 8: SMART WATER METER PROTOTYPE FINAL PROGRAM

On the final week, we embarked on quest to join all the component programs developed throughout the weeks to achieve a final single program for the Smart Water Meter Project Prototype. The final code included STM32F411CEU6 + HT1621 + Water Floe Sensor + GSM SIM7600CE + Motorized Ball Valve + IR Proximity Sensor + Power LED. The EEPROM was not included at the moment because it had not been delivered for testing. The final program compiled with Zero Errors and Zero Warnings. The Execution Time was 1s.747ms. The entire program had 426 Lines of Code. The final program is pasted below.

Smart Water Meter Prototype Final Program

```
                        SMART-WATER-METER PROJECT (c) ENG. ERIC MULWA
******************************************************************************************************

  * @file: main.c                  426 Lines of Code
  * @brief: Main program body

******************************************************************************************************

COPYRIGHT (C) ENG. ERIC MULWA
ALL RIGHTS RESERVED.

THIS PROJECT IS THE WORK OF ERIC MULWA DONE AT CDED CENTER, DEKUT, DURING HIS FINAL YEAR EXTERNAL
ATTACHMENT.

THE PROJECT IS A SMART-WATER-METER THAT CAN READ THE VOLUME OF WATER CONSUMED PER DAY IN M^3, DISPLAY THE
VOLUME IN THE 7-SEGMENT DISPLAY (HT1621) IN REAL TIME AND SEND THE DATA WIRELESSLY THROUGH AN SMS TO THE
USER & TO THE WATER PROVIDER SERVERS FOR BILLING.

******************************************************************************************************

/* USER CODE END Header */
/* Includes ------------------------------------------------------------------------------------------
----*/
#include "main.h"
/* Private includes ----------------------------------------------------------------------------------
----*/
/* USER CODE BEGIN Includes */
#include "stdio.h"
#include <string.h>
#include "HT1621.h"
/* USER CODE END Includes */

/* Private variables ---------------------------------------------------------------------------------
----*/
SPI_HandleTypeDef hspi2;

UART_HandleTypeDef huart1;

/* USER CODE BEGIN PV */
const uint8_t displ_size = 4;

//Motor Control with GSM section 1
#define PREF_SMS_STORAGE "\"SM\""
char ATcommand[80];
uint8_t ATisOK = 0;
uint8_t slot = 0;
```

```
uint8_t rx_buffer[100] = {0};
uint8_t rx_index = 0;
uint8_t rx_data;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------------------------
---*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_SPI2_Init(void);
static void MX_USART1_UART_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------------------------
---*/
/* USER CODE BEGIN 0 */
volatile uint32_t pulseCount = 0; //Read interrupt pulses from the Pin

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{

  /* MCU Configuration--------------------------------------------------------------------------
----*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_SPI2_Init();
  MX_USART1_UART_Init();
  /* USER CODE BEGIN 2 */
  HT1621_Init(displ_size);
  HT1621_Clear();
  HT1621_String("HELO", 9);
  HAL_Delay(1000);

  //Motor control with GSM section 2
  // Wait until getting response OK to AT
  while(!ATisOK)
  {
    sprintf(ATcommand,"AT\r\n");
    HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
    HAL_UART_Receive (&huart1, rx_buffer, 100, 100);
    HAL_Delay(1000);
    if(strstr((char *)rx_buffer,"OK"))
    {
      ATisOK = 1;
    }
    HAL_Delay(1000);
    memset(rx_buffer,0,sizeof(rx_buffer));
  }

  // Send AT+CMGF=1
  sprintf(ATcommand,"AT+CMGF=1\r\n");
  HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
  HAL_UART_Receive (&huart1, rx_buffer, 100, 100);
  HAL_Delay(1000);
  memset(rx_buffer,0,sizeof(rx_buffer));
```

```
  // Send AT+CNMI=2,1 to enable notification when SMS arrives
  sprintf(ATcommand,"AT+CNMI=2,1\r\n");
  HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
  // Enabling interrupt receive
  HAL_UART_Receive_IT(&huart1,&rx_data,1);// receive data (one character only)
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
//Display the water volume in M^3 consumed by the user
          float Volume = pulseCount * 0.003165;
          HT1621_Digit(Volume, 0, false, 6);
          HAL_Delay(500);
//Sending Volume of water Consumed SMS
          char mobileNumber1[] = "+254796456877";  // User end mobile number.
          char mobileNumber2[] = "+254796456877";  // Office end mobile number.
          char ATcommand[80];
          uint8_t buffer[30] = {0};
          uint8_t ATisOK = 0; //unsigned 8-bit integer variable named ATisOK, initialized to 0.
          //loop that will keep running until ATisOK becomes true.
          while(!ATisOK){
                        sprintf(ATcommand,"AT\r\n");
                        HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
                        HAL_UART_Receive (&huart1, buffer, 30, 100);
                        HAL_Delay(1000); //Gives GSM module enough time to respond to the AT command.
                        if(strstr((char *)buffer,"OK")){
                                ATisOK = 1;
                        }
                        HAL_Delay(1000);
                        memset(buffer,0,sizeof(buffer)); // clears the buffer variable
          }
          sprintf(ATcommand,"AT+CMGF=1\r\n"); //sets the GSM module to text mode.
          HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000); //sends the ATcommand to
the GSM
          HAL_UART_Receive (&huart1, buffer, 30, 100); //receives response from GSM module and store in
the buffer variable
          HAL_Delay(1000);
          memset(buffer,0,sizeof(buffer));
          //send data to user end
          sprintf(ATcommand,"AT+CMGS=\"%s\"\r\n",mobileNumber1); //sets phone number to send SMS
          HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
          HAL_Delay(100);
      sprintf(ATcommand,"Volume of Water consumed is: %.2f%c, Liters", Volume, 0x1a); // format the
message with the value of Volume
      HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
      HAL_UART_Receive (&huart1, buffer, 30, 100);
      memset(buffer,0,sizeof(buffer));
      //send data to office end
          sprintf(ATcommand,"AT+CMGS=\"%s\"\r\n",mobileNumber2); //sets phone number to send SMS
          HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
          HAL_Delay(100);
      sprintf(ATcommand,"Volume of Water consumed is: %.2f%c, Liters", Volume, 0x1a); // format the
message with the value of Volume
      HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
      HAL_UART_Receive (&huart1, buffer, 30, 100);
      memset(buffer,0,sizeof(buffer));
      HAL_Delay(86400000); //delay for 24 hours

  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
```

```c
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 12;
  RCC_OscInitStruct.PLL.PLLN = 96;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_3) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief SPI2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SPI2_Init(void)
{

  /* SPI2 parameter configuration*/
  hspi2.Instance = SPI2;
  hspi2.Init.Mode = SPI_MODE_MASTER;
  hspi2.Init.Direction = SPI_DIRECTION_2LINES;
  hspi2.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi2.Init.CLKPolarity = SPI_POLARITY_HIGH;
  hspi2.Init.CLKPhase = SPI_PHASE_2EDGE;
  hspi2.Init.NSS = SPI_NSS_SOFT;
  hspi2.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_256;
  hspi2.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi2.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi2.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi2.Init.CRCPolynomial = 10;
  if (HAL_SPI_Init(&hspi2) != HAL_OK)
  {
    Error_Handler();
  }
}
```

```
/**
  * @brief USART1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART1_UART_Init(void)
{

  /* USER CODE END USART1_Init 1 */
  huart1.Instance = USART1;
  huart1.Init.BaudRate = 115200;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart1) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(HT1621_CS_GPIO_Port, HT1621_CS_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : HT1621_CS_Pin */
  GPIO_InitStruct.Pin = HT1621_CS_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(HT1621_CS_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : Pulses_Pin */
  GPIO_InitStruct.Pin = Pulses_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(Pulses_GPIO_Port, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI9_5_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI9_5_IRQn);

}

/* USER CODE BEGIN 4 */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
  if(huart->Instance==USART1)
  {
    // if the character received is other than 'enter' ascii13, save the data in buffer
    if(rx_data!=13)
    {
      rx_buffer[rx_index++]=rx_data;
    }
```

```
    else
    {
      // if new message arrived, read the message
      if( sscanf((char*)rx_buffer, "\n+CMTI: " PREF_SMS_STORAGE ",%hhd", &slot)==1)
      {
        sprintf(ATcommand,"AT+CMGR=%d\r\n",slot);
        HAL_UART_Transmit_IT(&huart1,(uint8_t *)ATcommand,strlen(ATcommand));
      }
      // if message read contains "valve-on", Open the ball valve for water to flow
      else if (strstr((char *)rx_buffer,"Valve-on"))
      {
          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
          HAL_Delay(2800);
          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
          HAL_Delay(2000);
      }
      // if message read contains "valve-off", close the ball valve to cut water supply
      else if (strstr((char *)rx_buffer,"valve-off"))
      {
          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);
          HAL_Delay(2800);
          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
          HAL_Delay(2000);
      }
      // This will delete all messages in the SIM card.
      if (strstr((char *)rx_buffer,"READ"))
      {
          sprintf(ATcommand,"AT+CMGD=,4\r\n");
          HAL_UART_Transmit_IT(&huart1,(uint8_t *)ATcommand,strlen(ATcommand));
      }
      rx_index=0;
      memset(rx_buffer,0,sizeof(rx_buffer));
    }
    // Enabling interrupt receive again
    HAL_UART_Receive_IT(&huart1,&rx_data,1); // receive data (one character only)
  }
}
/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}
```

PROBLEMS ENCOUNTERED

Some of the problems encountered while developing the project include;

- Limited knowledge of embedded systems – The Smart Water Meter Project required a good understanding of embedded systems and as a student of Electrical Engineering I was very challenged. I had to start from scratch and do a lot of reading before I could complete a working code. I also encountered challenges in understanding the various components and how they interacted with each other at the beginning of the attachment.
- Programming – Programming the microcontroller (STM32F411CEU6) using the embedded C language challenged me since I had not been exposed to programming before. Writing efficient and optimized code that consumes low power while performing the required functions required a steep learning curve.
- Sensor Selection and Calibration – Selecting the right sensors that meet the requirements of the project was also challenging. I encountered difficulties in calibrating the sensors to ensure accurate measurement of water flow rate and other parameters.
- Wireless Communication – Implementing wireless communication protocols using the GSM SIM7600CE was a big challenge. As a matter of fact, it was my first time to hear and see GSM module. I had limited knowledge of wireless communication systems and I was forced to do a lot of research to actualize the project.
- Limited Budget – Developing a smart water meter project requires significant financial resources, which are a challenge for students working on a limited budget. I to find creative ways to minimize project costs while ensuring that the project met the required specifications.

CONCLUSION

In conclusion, the development of a smart water meter project using the STM32F411CEU6 microcontroller was a challenging but rewarding experience. The project required a good understanding of embedded systems, programming, sensor selection and calibration, hardware design, and wireless communication. Despite these challenges, I was able to design and develop a functional smart water meter that accurately measured water flow rate and transmitted data wirelessly to a remote server. The project demonstrated my ability to apply engineering principles and knowledge to solve real-world problems. Future improvements could include integrating more advanced sensor technology and implementing more efficient power-saving techniques. Overall, the project was a success and provided valuable experience for me and the group members. Having completed my eight weeks attachment at CDED, I am proud to say that I have gained a lot of skills and experiences that will enable me to work effectively as far as Embedded systems and PCB design is concerned. I am therefore confident that CDED center has given me the skills and experience I require to venture into the Electronics job market.

RECOMMENDATIONS

I would like to make the following recommendations to Center for Development of Electronic Devices about the Smart Water Meter Project Developed;

- The smart water meter project can be enhanced by implementing a more advanced algorithm for analyzing data collected from the sensors. This can provide additional insights into water usage patterns and help identify leaks and other issues in the water distribution system.
- Increase the battery life – to improve the efficiency of the smart water meter, efforts can be made to reduce the power consumption of the device, thereby extending its battery life. This can be achieved through the use of low-power components, optimization of the firmware, and improved power management techniques.
- Improve wireless communication – to enhance the reliability of the wireless communication between the smart water meter and the remote server, more robust and efficient wireless communication protocols can be used, such as LoRaWAN, NB-IoT or LTE-M.
- Conduct field testing: It is recommended to conduct field testing of the smart water meter project to validate its performance in real-world conditions. This can help identify any issues and provide valuable feedback for further improvements.
- Consider user feedback – it is important to consider user feedback when designing smart water meter projects. Incorporating feedback from end-users can help identify areas for improvement and ensure that the final product meets the needs and expectations of its intended users.

REFERENCES

Alvisi, S., Casellato, F., Franchini, M., Govoni, M., Luciani, C., Poltronieri, F., ... & Tortonesi, M. (2019). Wireless middleware solutions for smart water metering. *Sensors*, *19*(8), 1853. https://sdbindex.com/documents/00000002/00000-48435.pdf

Li, X. J., & Chong, P. H. J. (2019). Design and implementation of a self-powered smart water meter. *Sensors*, *19*(19), 4177. https://www.mdpi.com/542064

March, H., Morote, Á. F., Rico, A. M., & Saurí, D. (2017). Household smart water metering in Spain: Insights from the experience of remote meter reading in alicante. *Sustainability*, *9*(4), 582. https://www.mdpi.com/191074

Muhammetoglu, A., Albayrak, Y., Bolbol, M., Enderoglu, S., & Muhammetoglu, H. (2020). Detection and assessment of post meter leakages in public places using smart water metering. *Water Resources Management*, *34*, 2989-3002. https://link.springer.com/article/10.1007/s11269-020-02598-1

Public Utilities Board Singapore Pan_Ju_Khuan@ pub. gov. sg. (2016). Managing the water distribution network with a Smart Water Grid. *Smart Water*, *1*(1), 4. https://link.springer.com/article/10.1186/s40713-016-0004-4

Suresh, M., Muthukumar, U., & Chandapillai, J. (2017, July). A novel smart water-meter based on IoT and smartphone app for city distribution management. In *2017 IEEE region 10 symposium (TENSYMP)* (pp. 1-5). IEEE. https://ieeexplore.ieee.org/abstract/document/8070088/

Živic, N. S., Ur-Rehman, O., & Ruland, C. (2015, November). Evolution of smart metering systems. In *2015 23rd telecommunications forum telfor (TELFOR)* (pp. 635-638). IEEE. https://ieeexplore.ieee.org/abstract/document/7377547/

APPENDIX

*"Words cannot express accurately how happy I am to achieve a **Working Prototype** of the **Smart Water Meter Project**, having started from scratch without knowing anything about water meters. It was a breakthrough in my career as an **Electrical and Electronics Engineer** as far as **Embedded Systems** are concerned. I couldn't be prouder of myself."*

Eric Muinde Mulwa, E021-01-0694/2019, BSc Electrical and Electronics Engineering, DeKUT.