COMPUTERS IN CONTROL NME3545

Sylvester Chiamaka Azubuine (U2075835)

Fuel Delivery System

Submitted in partial fulfilment of

the requirements of University of Huddersfield

for the Degree of MSc Engineering Control Systems and Instrumentation

School of Computing & Engineering

April 2021

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

ARM:	Advanced RISC Machine
RISC:	Reduced Instruction Set for Computer processors
IoT:	Internet of Things
IDE:	Integrated Development Environment
LCD:	Liquid Crystal Display
LED:	Light emitting diode
PB:	Push Button
IND:	Indicator
I/0:	Input Output
SPI:	Serial Port Interface
SDK:	Software Development Kit
RTOS:	Real-time Operating System
GPIO:	General-purpose input/output
RTC:	Real-time clock
ADC:	Analog-to-digital converter

ACKNOWLEDGEMENT

I would like to acknowledge the support my module tutors, Dr Maryam Hafeez and Dr. Violeta Holmes provided me during the execution of this assignment. I specifically want to thank Dr. Maryam for her encouragement when I was struggling with understanding C programming language.

I also want to thank my personal academic tutor Prof. Andrew Longstaff, and the library staff Dr. Alison Sharman for their guidance on the use of the library resources while I was writing this report.

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Birds Eye View

An embedded system is basically a computer system included within an equipment/device with a specific function, and mostly executes this function with real-time computing constraints in size, power, memory, computation, and cost (Toulson, 2017). This type of system is built around a specific task and therefore only includes hardware and software that are relevant to the task. The failure/malfunction of an embedded system can be life threatening and therefore must comply with some important safety certifications (Toulson & Wilmshurst, 2016). Its implementations can be found in all aspects of the modern life such as, consumer electronics (Television & refrigerators), flight control systems, automotive machines, commerce and logistics, and industrial application (Iniewski, 2012).

An important use case of embedded system can be seen in the fuel delivery system at modern petrol stations. This is implemented by interfacing transducers (sensors) and actuators (electric motors) to a micro-controller using analogue converters, digital input/output ports and an LCD display. (Hafeez, 2021).

This system is being developed using the technology provided by ARM Limited, a British semiconductor and software design company based in Cambridge, England. This company was founded in 1990 and develops 32- and 64-bit processors meant for high end applications, factoring in the performance, power, and cost requirements of its intended use (Ibrahim, 2019). Its application ranges from small, interconnected devices (IoT) to robust and advanced units like autonomous vehicles (Arm Limited, 2021a).

Embedded Systems can be classified into three major types.

- The stand-alone system: This type of system performs its function without the aid of another system. A washing machine is an example of this type of system.
- Real time embedded system: This system monitors and responds to its immediate external environment in real time. Examples includes autonomous vehicles, streetlights, and air conditioning systems.
- Network embedded System: This type has a distributed computing environment consisting of smaller units working together. This type of system gives rise to the IoT. Smartphones, smartwatches, and laptop pcs falls into this category because they rely on a network of more connected devices to function optimally.
 (Johnson & Hafeez, 2021)

The system being developed in this assignment will be a stand-alone system consisting of a software and hardware, with the sole task of delivering metered fuel to a paying customer.

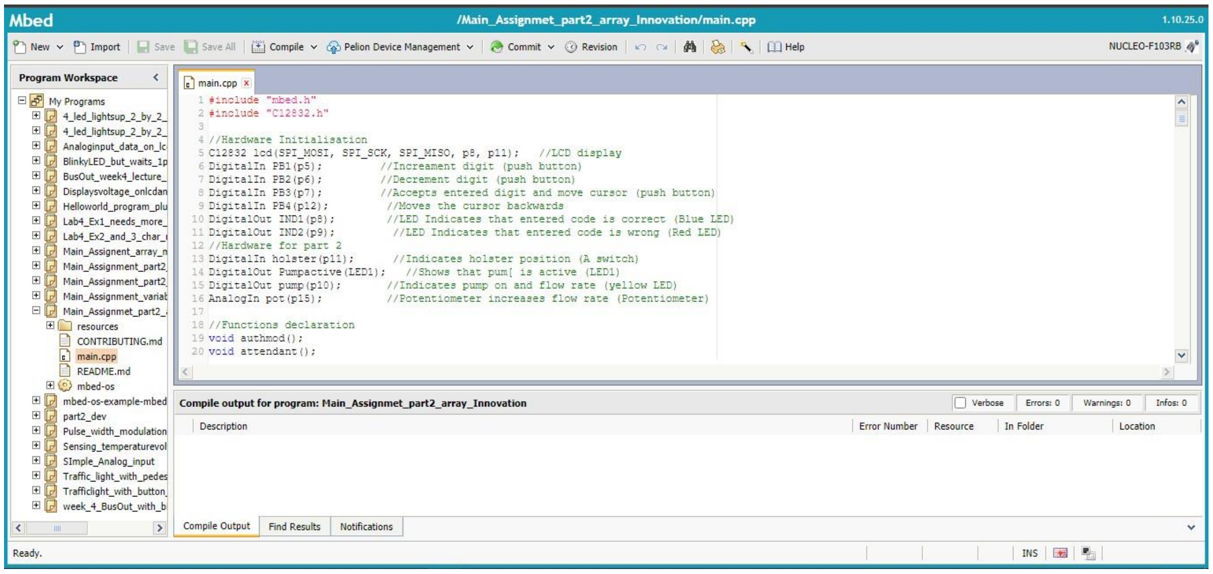
1.2 Objectives

- To write a structured and annotated ‘C’ program using the mbed compiler.
- To provide evidence of the development of each stage of the program.
- To include some improvements to the implementation of the system.
- To demonstrate a complete working program to the tutor in a laboratory session.

2 LITERATURE REVIEW

2.1 Software Specification

The software that runs on Cortex-M microprocessors are developed using the ARM mbed platform. This platform features a Software Development Kit (SDK) made up of C/C++ software libraries such as peripheral drivers, networking, RTOS and runtime environment. The mbed IDE on the platform, provides an online environment where a program can be written and compiled, making use of the available SDK. It features a huge library of APIs accessible to anyone at no cost. The online mbed simulator as shown in figure 2, enables the simulation of the development boards as well as some peripheral hardware which are added to the right of the screen while the accompanying program is written on the left pane. The compiler shown in figure 1, enables a completed source code to be translated into a binary machine code that can be downloaded into a physical development board for execution (Johnson & Hafeez, 2021).



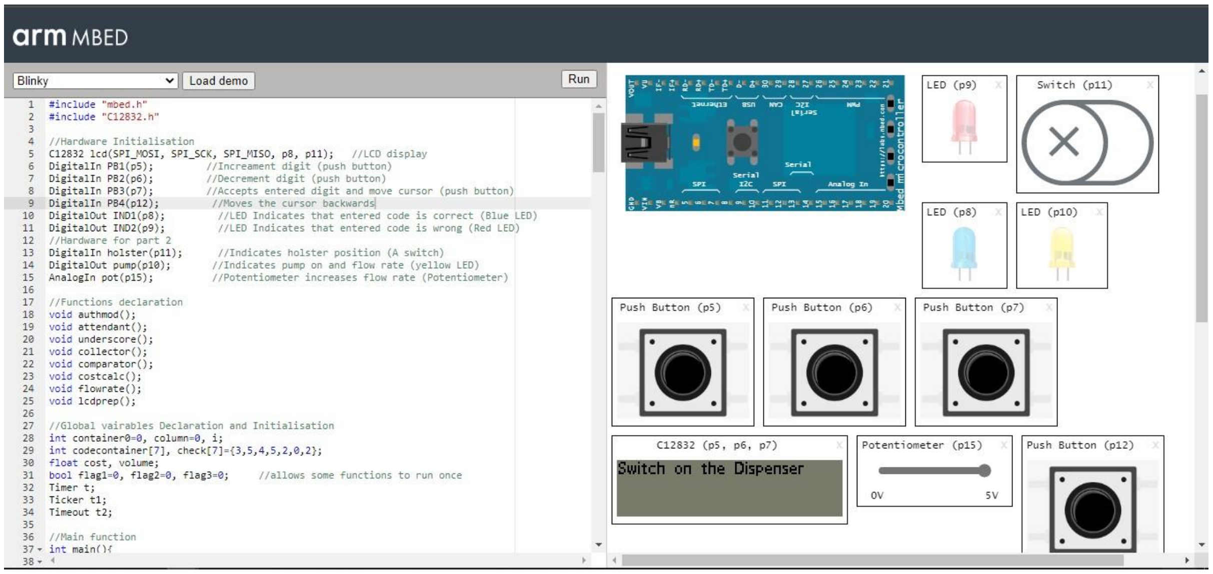
```
#include "mbed.h"
#include "C12832.h"

//Hardware Initialisation
C12832 lcd(SPI_MOSI, SPI_SCK, SPI_MISO, p8, p11);   //LCD display
DigitalIn PB1(p5);         //Increament digit (push button)
DigitalIn PB2(p6);         //Decrement digit (push button)
DigitalIn PB3(p7);         //Accepts entered digit and move cursor (push button)
DigitalIn PB4(p12);         //Moves the cursor backwards
DigitalOut IND1(p8);        //LED Indicates that entered code is correct (Blue LED)
DigitalOut IND2(p9);         //LED Indicates that entered code is wrong (Red LED)
//Hardware for part 2
DigitalIn holster(p11);      //Indicates holster position (A switch)
DigitalOut Pumpactive(LED1);   //Shows that pum[ is active (LED1)
DigitalOut pump(p10);       //Indicates pump on and flow rate (yellow LED)
AnalogIn pot(p15);          //Potentiometer increases flow rate (Potentiometer)

//Functions declaration
void authmod();
void attendant();
```

Compile output for program: Main_Assignmet_part2_array_Innovation

Verbose | Errors: 0 | Warnings: 0 | Infos: 0

Description	Error Number	Resource	In Folder	Location

Figure 1: Mbed IDE (Arm Limited, 2021).



```
#include "mbed.h"
#include "C12832.h"

//Hardware Initialisation
C12832 lcd(SPI_MOSI, SPI_SCK, SPI_MISO, p8, p11);   //LCD display
DigitalIn PB1(p5);         //Increament digit (push button)
DigitalIn PB2(p6);         //Decrement digit (push button)
DigitalIn PB3(p7);         //Accepts entered digit and move cursor (push button)
DigitalIn PB4(p12);         //Moves the cursor backwards
DigitalOut IND1(p8);         //LED Indicates that entered code is correct (Blue LED)
DigitalOut IND2(p9);         //LED Indicates that entered code is wrong (Red LED)
//Hardware for part 2
DigitalIn holster(p11);      //Indicates holster position (A switch)
DigitalOut pump(p10);       //Indicates pump on and flow rate (yellow LED)
AnalogIn pot(p15);          //Potentiometer increases flow rate (Potentiometer)

//Functions declaration
void authmod();
void attendant();
void underscore();
void collector();
void comparator();
void costcalc();
void flowrate();
void lcdprep();

//Global vairables Declaration and Initialisation
int container0=0, column=0, i;
int codecontainer[7], check[7]={3,5,4,5,2,0,2};
float cost, volume;
bool flag1=0, flag2=0, flag3=0;     //allows some functions to run once
Timer t;
Ticker t1;
Timeout t2;

//Main function
int main(){
```

Figure 2: Mbed simulator (Arm Limited, 2021).

Some of the libraries provided on the ARM Mbed platform used in developing this system are listed in table 1 below.

S/N	Library	Function
1	Analog I/O	This library provides API for interfacing analog hardware devices such as potentiometer.
2	Digital I/O	This library provides API for interfacing digital input or output devices such as LEDs and switches
4	SPI	This library provides API for interfacing with serial peripheral devices such as LCDs
5	Timers	This library provides API for the implementing of time-based functions, such as timeout and ticker functions
6	wait	This library also implements a time-based function useful when working with LEDs
7	InterruptIn	Provides an API for triggering an event when a digital input pin changes

Table 1: mbed libraries (Arm Limited, 2021b).

2.2 Hardware Specification

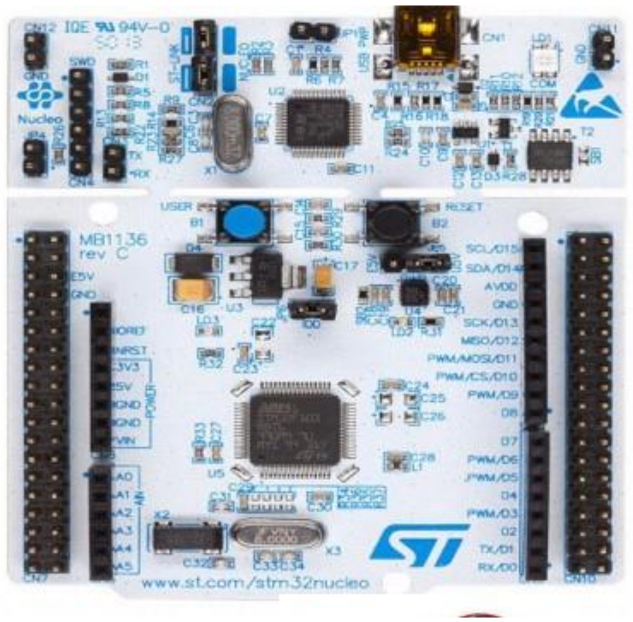


Figure 3: Nucleo STM32F103 (Johnson & Hafeez, 2021)

ARM supports a Hardware Development Kit (HDK) that features some off-the-shelf development boards to support a program development. These boards provide a fast way to get started on the mbed platform because of its low cost and support for flexible rapid prototyping (Johnson & Hafeez, 2021). ARM supports three types of microcontroller-based hardware, the Cortex A, R, and M. Cortex M is the lowest in power consumption and is cost friendly. It supports a 32-bit RISC architecture, Harvard architecture, a 3-stage branch speculation pipeline and interrupts (Ibrahim, 2019). The Nucleo-F103RB microcontroller, is a part of the Cortex M hardware family, and the major hardware supporting this system. Its features are shown in table 2 below.

S/N	Features
1	ARM32-bit Cortex-M3 CPU
2	72 MHz max CPU frequency
3	VDD from 2.0 V to 3.6 V
4	128 KB Flash
5	GPIO with external interrupt capability
6	12-bit ADC with 16 channels
7	RTC
8	Timers
9	I2C
10	SPI
11	USB 2.0 full speed

Table 2: Nucleo STM32F103 features (Arm Limited, 2021c).

3 DESIGN/IMPLEMENTATION

3.1 System Architecture of the Dispenser

The design of this system which includes the requirements and specifications, as well as the program that runs on the micro-controller was performed in line with the ISO/IEC/IEEE 29148:2011 standard that defines a project (IEEE, 2021).

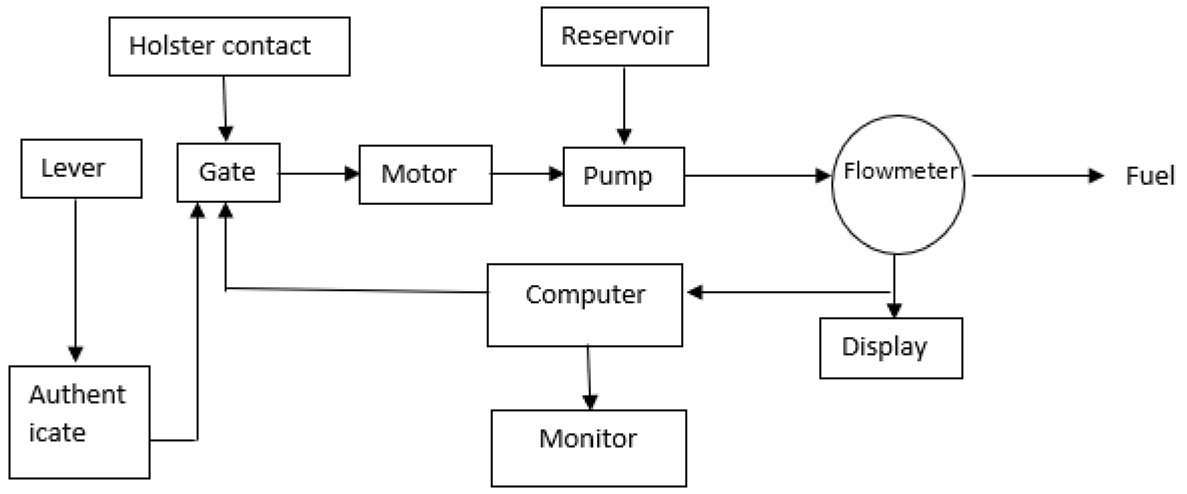


Figure 4: Block diagram of the fuel delivery system (Hafeez, 2021).

The system was designed in two parts.

In Part 1, the authentication block was developed using an LCD, pushbuttons, and LEDs as shown in figure 2. A 'C' program was written to emulate an authentication sub-system and its operation was evaluated using the Nucleo-F103RB microcontroller board simulated on the ARM mbed platform/IDE (Hafeez, 2021).

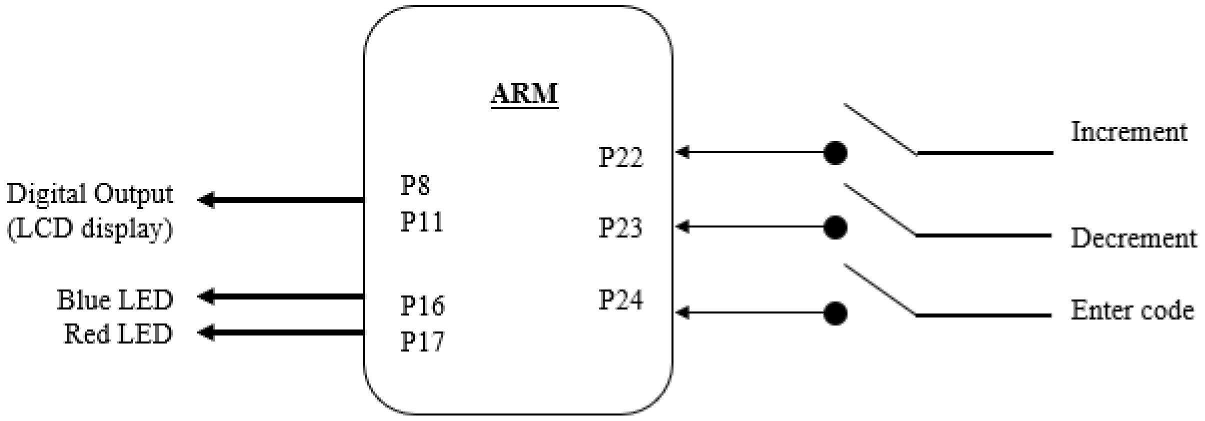


Figure 5: I/O interfacing for authentication (Hafeez, 2021).

In part 2, the dispenser's operation was developed using a switch representing the holster contact, LEDs indicating the flow rate and a potentiometer representing the pressure lever on the nozzle of a fuel dispenser (Hafeez, 2021).
The authentication block developed in part 1 was merged with the part 2 resulting in a complete fuel delivery system.

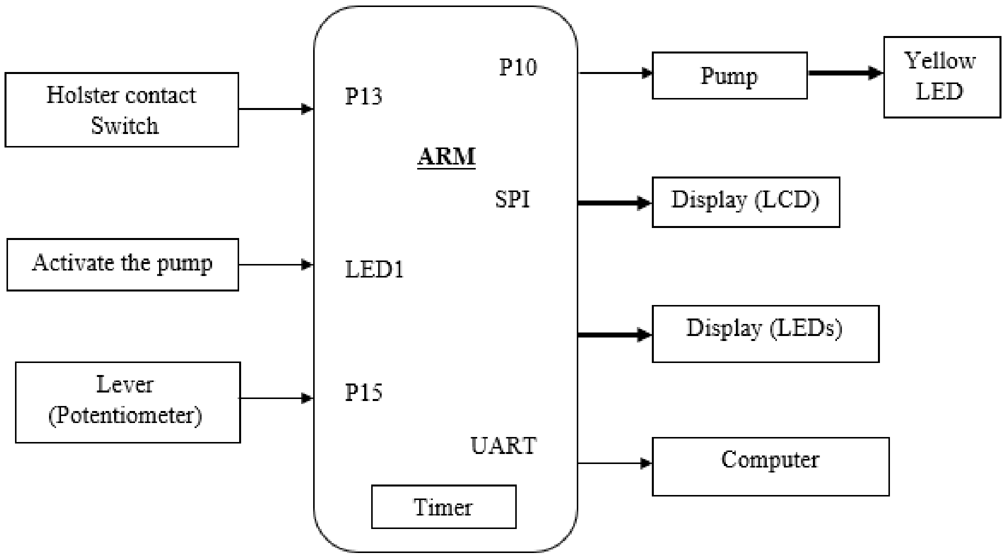


Figure 6: Block diagram of the micro-controller interface for the dispenser (Hafeez, 2021).

3.1.1 SYSTEM SPECIFICATIONS

S/N	Requirements	Specifications
1	Holster	Switch on/off positions, indicating position of nozzle (p11).
2	Display	C12832 lcd (SPI_MOSI, SPI_SCK, SPI_MISO, p8, p11)
3	Numeric Keypad	3 push buttons (p5, p6, p7)
4	Indicators	LEDs (Red-p9, Blue-p8, and onboard LEDs)
5	Pump	Yellow LED – (p10)
6	Nozzle	
7	Lever on the Nozzle	5v Potentiometer (p15)
8	Serial output port	Connects to the cashier's pc monitor
9	Flowmeter	Timer Function

Table 3: System Requirements.

S/N	Constraints	Values
1	Dispenser Power	2.2KW
2	Voltage	240V
3	Highest flow rate	1 litre/second
4	Maximum fuel delivery cost	£150
4	Passcode	7 digits
5	Fuel metering cost	£1.5 per litre
6	Highest potentiometer voltage	5V

Table 4: Constraints.

3.2 SYSTEM DESIGN AND ANALYSIS

3.2.1 Code Structure: Flow chart

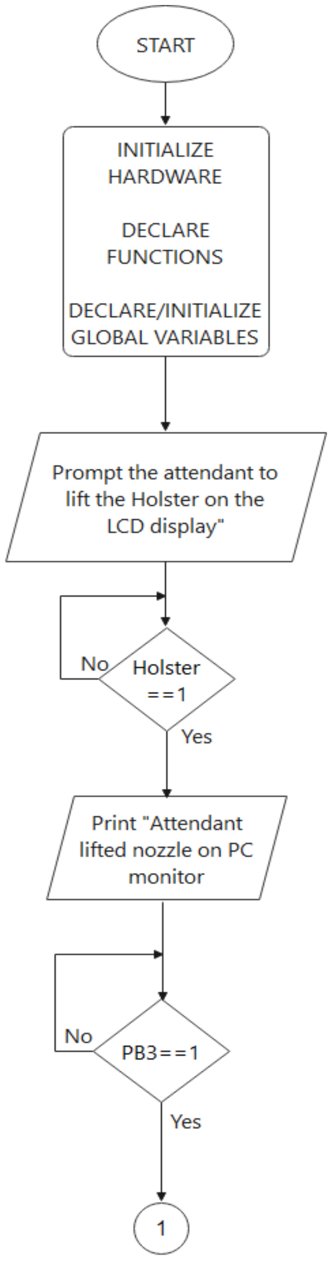

START
INITIALIZE HARDWARE
DECLARE FUNCTIONS
DECLARE/INITIALIZE GLOBAL VARIABLES
Prompt the attendant to lift the Holster on the LCD display"
Holster ==1
No
Yes
Print "Attendant lifted nozzle on PC monitor
PB3==1
No
Yes
1

Figure 7: Declarations and prompt to lift the Holster contact.

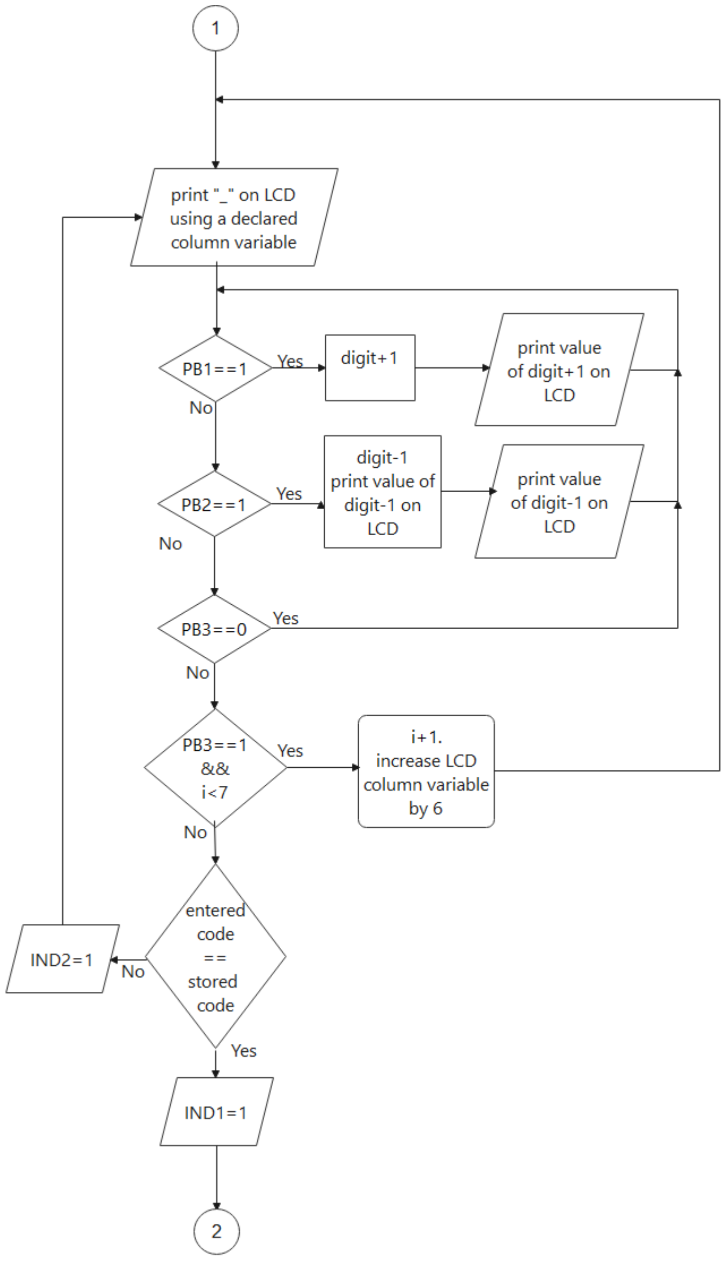

1
print "_" on LCD using a declared column variable
PB1==1
Yes
No
digit+1
print value of digit+1 on LCD
PB2==1
Yes
No
digit-1 print value of digit-1 on LCD
print value of digit-1 on LCD
PB3==0
Yes
No
PB3==1 && i<7
Yes
No
i+1. increase LCD column variable by 6
entered code == stored code
No
Yes
IND2=1
IND1=1
2

Figure 8: Authentication of passcode.

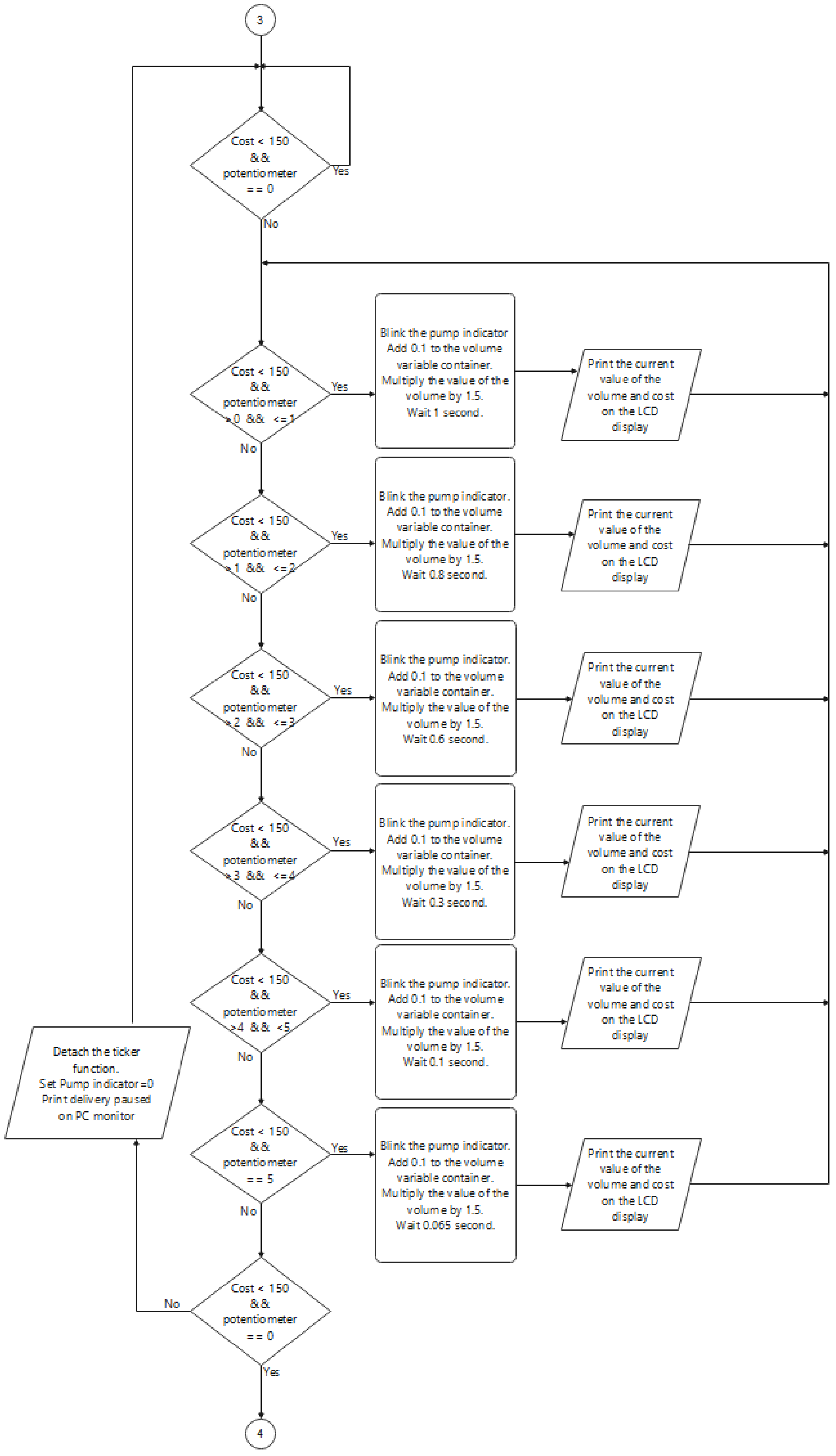

3
Cost < 150 && potentiometer == 0
Yes
No
Cost < 150 && potentiometer >0 && <=1
Yes
No
Blink the pump indicator Add 0.1 to the volume variable container. Multiply the value of the volume by 1.5. Wait 1 second.
Print the current value of the volume and cost on the LCD display
Cost < 150 && potentiometer >1 && <=2
Blink the pump indicator. Add 0.1 to the volume variable container. Multiply the value of the volume by 1.5. Wait 0.8 second.
Cost < 150 && potentiometer >2 && <=3
Blink the pump indicator. Add 0.1 to the volume variable container. Multiply the value of the volume by 1.5. Wait 0.6 second.
Cost < 150 && potentiometer >3 && <=4
Blink the pump indicator. Add 0.1 to the volume variable container. Multiply the value of the volume by 1.5. Wait 0.3 second.
Cost < 150 && potentiometer >4 && <5
Blink the pump indicator. Add 0.1 to the volume variable container. Multiply the value of the volume by 1.5. Wait 0.1 second.
Detach the ticker function. Set Pump indicator=0 Print delivery paused on PC monitor
Cost < 150 && potentiometer == 5
Blink the pump indicator. Add 0.1 to the volume variable container. Multiply the value of the volume by 1.5. Wait 0.065 second.
Cost < 150 && potentiometer == 0
4

Figure 9: Fuel metering and cost calculation.

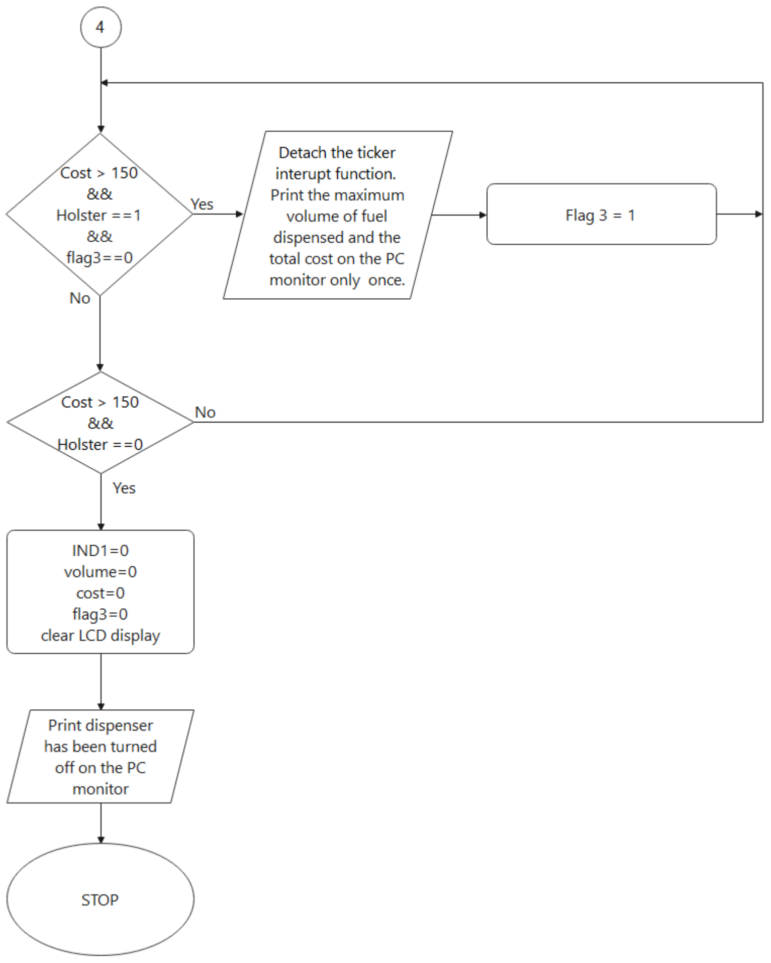

4
Cost > 150 && Holster ==1 && flag3==0
Yes
No
Detach the ticker interupt function. Print the maximum volume of fuel dispensed and the total cost on the PC monitor only once.
Flag 3 = 1
Cost > 150 && Holster ==0
No
Yes
IND1=0 volume=0 cost=0 flag3=0 clear LCD display
Print dispenser has been turned off on the PC monitor
STOP

Figure 10: Displaying result delivery result on Cashier's screen and ending the program.

3.2.3 Development Cycle and Testing:

The development, testing and debugging of the system were carried out on the ARM Mbed online simulator and follows the development cycle illustrated in figure 11 below.

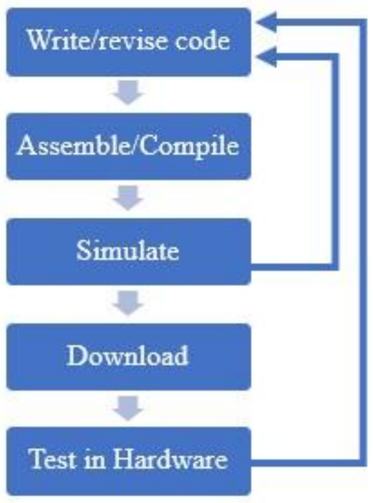


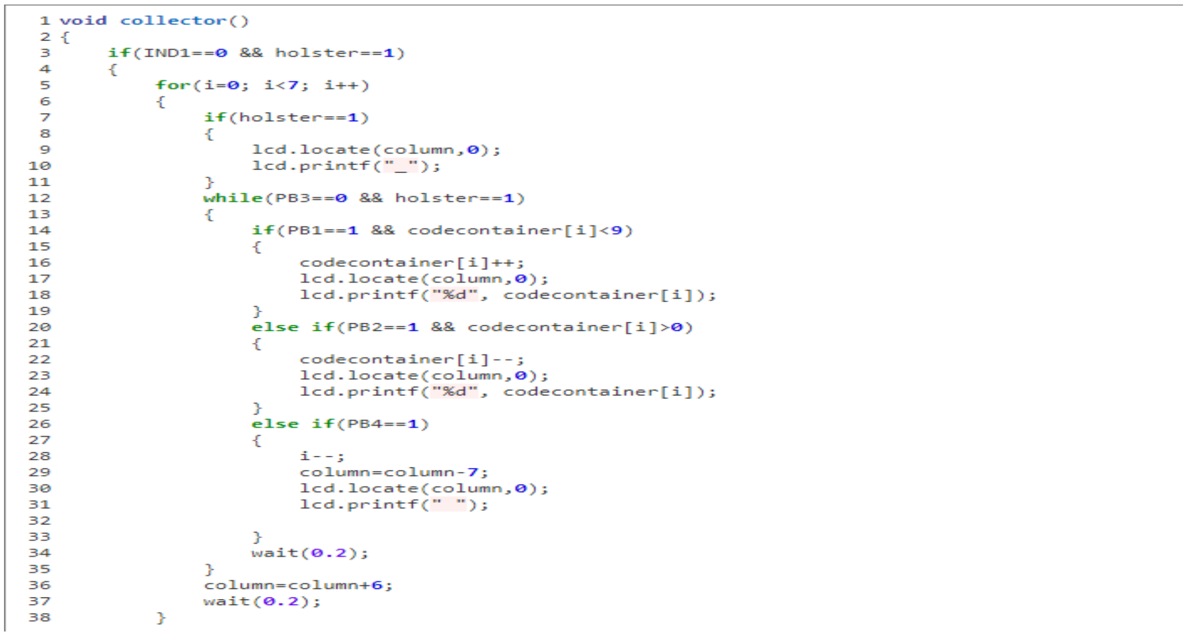
Figure 11: Program development cycle (Johnson & Hafeez, 2021).

3.2.2 Writing the Program.

The problem statement given in the assignment brief for the program was simplified into a software algorithm and flow chart, which was further broken down to subtasks that were written as a software code (Hafeez, 2021). The software code was written using an object-oriented programming language "C" (Vystavěl, 2017). Some of the subtasks created and written in this program are explained below:

Collecting the passcode from the attendant:

The collection of passcode from the user was implemented using a for loop as shown in figures 8 and 12. The loop circles back each time push button 3 was pressed and runs 7 times to collect the 7-digit passcode from the attendant. The attendant will be able to increment or decrement each digit using a set of push buttons.

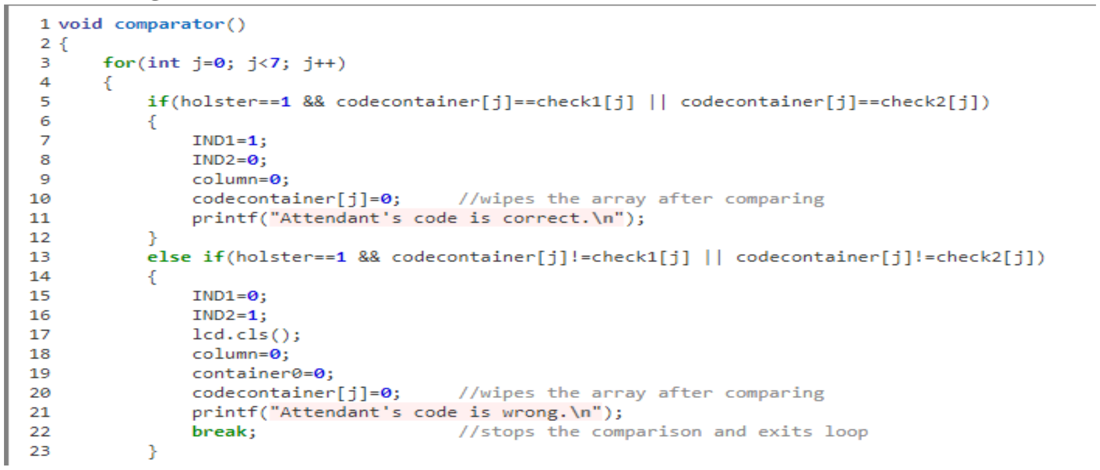


```c
void collector()
{
    if(IND1==0 && holster==1)
    {
        for(i=0; i<7; i++)
        {
            if(holster==1)
            {
                lcd.locate(column,0);
                lcd.printf("_");
            }
            while(PB3==0 && holster==1)
            {
                if(PB1==1 && codecontainer[i]<9)
                {
                    codecontainer[i]++;
                    lcd.locate(column,0);
                    lcd.printf("%d", codecontainer[i]);
                }
                else if(PB2==1 && codecontainer[i]>0)
                {
                    codecontainer[i]--;
                    lcd.locate(column,0);
                    lcd.printf("%d", codecontainer[i]);
                }
                else if(PB4==1)
                {
                    i--;
                    column=column-7;
                    lcd.locate(column,0);
                    lcd.printf(" ");

                }
                wait(0.2);
            }
            column=column+6;
            wait(0.2);
        }
```

Figure 12: Collector code snippet.

Comparing and authenticating the code:

The digits collected from the user were then compared with an already stored code in the program and when they match, the blue LED is turned on. This was also done using a for loop that scans through the codecontainer array and an if-else statement to compare the digits, as shown in figures 8 and 13.

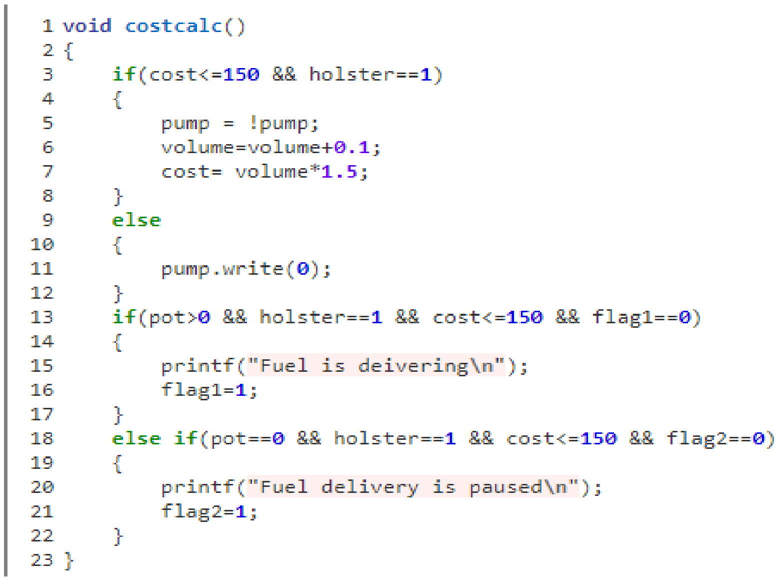


```c
void comparator()
{
    for(int j=0; j<7; j++)
    {
        if(holster==1 && codecontainer[j]==check1[j] || codecontainer[j]==check2[j])
        {
            IND1=1;
            IND2=0;
            column=0;
            codecontainer[j]=0;     //wipes the array after comparing
            printf("Attendant's code is correct.\n");
        }
        else if(holster==1 && codecontainer[j]!=check1[j] || codecontainer[j]!=check2[j])
        {
            IND1=0;
            IND2=1;
            lcd.cls();
            column=0;
            container0=0;
            codecontainer[j]=0;     //wipes the array after comparing
            printf("Attendant's code is wrong.\n");
            break;                  //stops the comparison and exits loop
        }
```

Figure 13: Comparator Code Snippet.

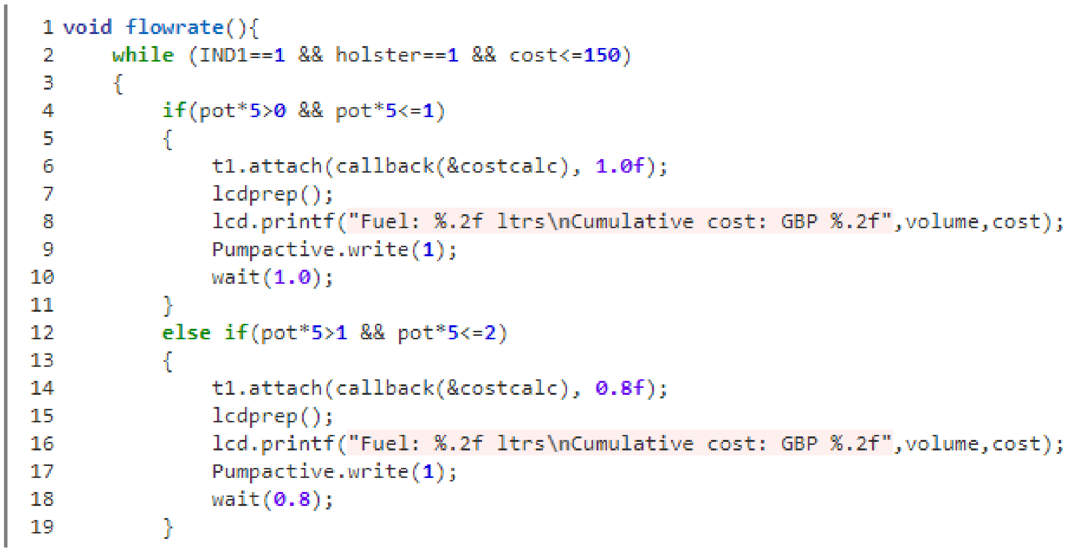
Metering the fuel:

The volume and cost of the fuel delivered was measured and calculated in the costcalc function shown in figure 14. The costcalc function attaches to the flowrate function at a varied time in seconds using a ticker statement, and the rate at which the costcalc function was being attached depends on the value from the potentiometer input as shown in figures 9 and 15.



```
void costcalc()
{
    if(cost<=150 && holster==1)
    {
        pump = !pump;
        volume=volume+0.1;
        cost= volume*1.5;
    }
    else
    {
        pump.write(0);
    }
    if(pot>0 && holster==1 && cost<=150 && flag1==0)
    {
        printf("Fuel is deivering\n");
        flag1=1;
    }
    else if(pot==0 && holster==1 && cost<=150 && flag2==0)
    {
        printf("Fuel delivery is paused\n");
        flag2=1;
    }
}
```

Figure 14: Calculating the volume and cost of the metered fuel.



```
void flowrate(){
    while (IND1==1 && holster==1 && cost<=150)
    {
        if(pot*5>0 && pot*5<=1)
        {
            t1.attach(callback(&costcalc), 1.0f);
            lcdprep();
            lcd.printf("Fuel: %.2f ltrs\nCumulative cost: GBP %.2f",volume,cost);
            Pumpactive.write(1);
            wait(1.0);
        }
        else if(pot*5>1 && pot*5<=2)
        {
            t1.attach(callback(&costcalc), 0.8f);
            lcdprep();
            lcd.printf("Fuel: %.2f ltrs\nCumulative cost: GBP %.2f",volume,cost);
            Pumpactive.write(1);
            wait(0.8);
        }
```

Figure 15: Snippet for calculating the fuel flow rate.

3.2.4 Discussion/Innovation

During the program development, there were some difficulties encountered on the Mbed simulation/compiler platform. The most prevalent issue was an unresponsive timer function. The ticker function was used instead to perform the volume calculation. Also, it was observed that the switch (holster) which represents the holster contact had to be pressed twice to turn it on. In addition, while entering the password using push buttons (PB1, PB2,PB3), the digits were often observed to continuously increment or decrement despite being pressed only once. This often caused the code entered to be incorrect.
Furthermore, Reusable blocks of code were made into a function that can be called into the main function or another sub function to further improve program adaptability. This became very useful as a previous authentication function/bock was easily swapped out with a simple and better written one during the program development.

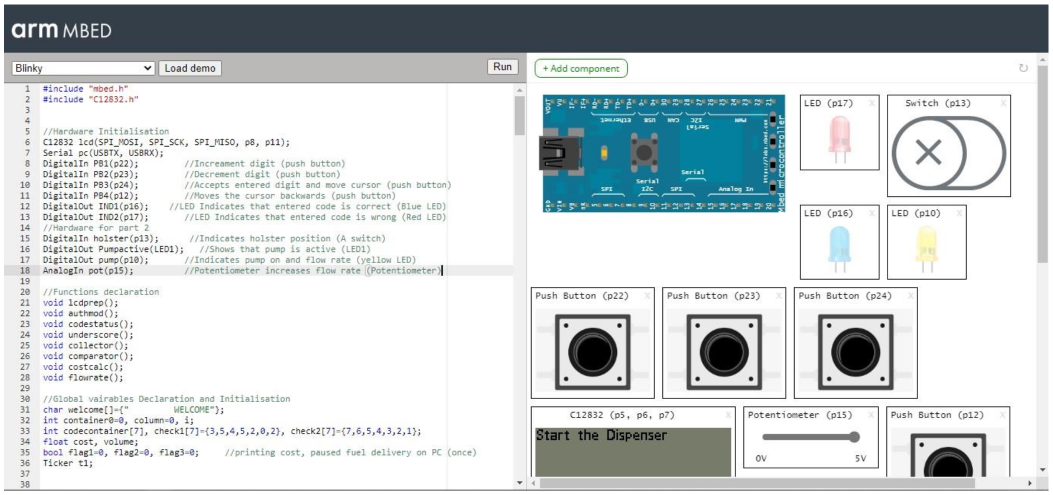
The innovations made to the program includes:

- Reusability of the fuel dispenser: This feature enables the dispenser to start afresh when it has dispensed its maximum fuel volume by turning the switch to an off/on position, representing the actions of the holster contact. This function also includes the ability to turn off the dispenser at any stage of its operation and turning it back on to start afresh with all variables reset to their initial values.
- The ability to authenticate a second attendant's passcode.
- The ability to re-authenticate any of the attendant's code if the passcode was wrong.
- The ability to shift back the cursor to edit an already entered code during the password authentication.

4. RECOMMENDATIONS AND CONCLUSIONS

The successful execution of this assignment demonstrates a clear understanding of the development methodology of an embedded system. This skill includes the understanding of the thought process that goes into the development of a standard 'C' program and can be adaptable to all other programming languages.
However, the inability to test the system on a hardware due to the covid -19 pandemic resulting in a shuttered laboratory, poses a substantial limitation in the skill/knowledge acquired. Hence, a practical implementation is recommended.
Finally, the implementation of embedded system is a task that can be performed by almost anyone who acquires the knowledge, due to the affordable hardware and software made available by ARM Ltd. Hence, encouraging creativity and delivering an entrepreneurial opportunity to the developer.

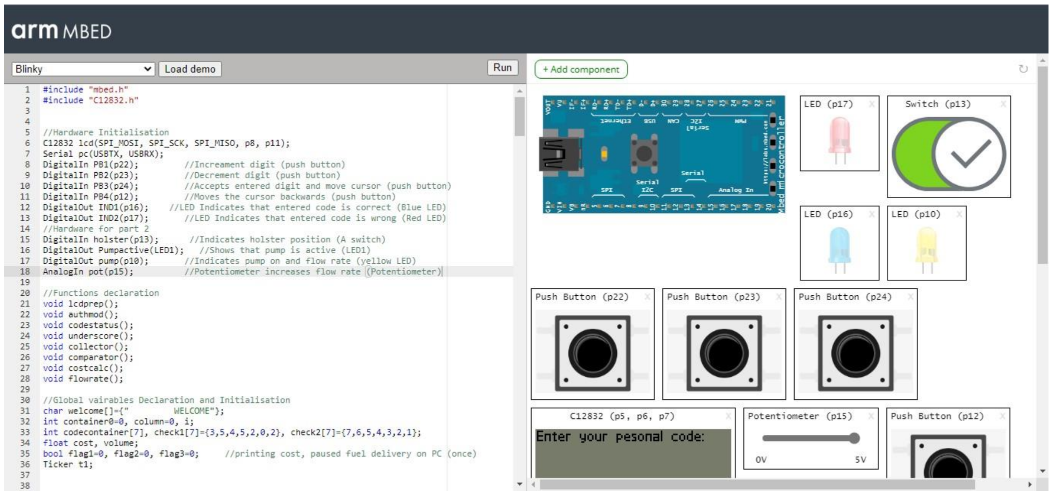
5. RESULTS FROM THE FUEL DELIVERY



```
#include "mbed.h"
#include "C12832.h"


//Hardware Initialisation
C12832 lcd(SPI_MOSI, SPI_SCK, SPI_MISO, p8, p11);
Serial pc(USBTX, USBRX);
DigitalIn PB1(p22);          //Increament digit (push button)
DigitalIn PB2(p23);          //Decrement digit (push button)
DigitalIn PB3(p24);          //Accepts entered digit and move cursor (push button)
DigitalIn PB4(p12);          //Moves the cursor backwards (push button)
DigitalOut IND1(p16);    //LED Indicates that entered code is correct (Blue LED)
DigitalOut IND2(p17);        //LED Indicates that entered code is wrong (Red LED)
//Hardware for part 2
DigitalIn holster(p13);       //Indicates holster position (A switch)
DigitalOut Pumpactive(LED1);   //Shows that pump is active (LED1)
DigitalOut pump(p10);        //Indicates pump on and flow rate (yellow LED)
AnalogIn pot(p15);           //Potentiometer increases flow rate (Potentiometer)

//Functions declaration
void lcdprep();
void authmod();
void codestatus();
void underscore();
void collector();
void comparator();
void costcalc();
void flowrate();

//Global vairables Declaration and Initialisation
char welcome[]={"         WELCOME"};
int container0=0, column=0, i;
int codecontainer[7], check1[7]={3,5,4,5,2,0,2}, check2[7]={7,6,5,4,3,2,1};
float cost, volume;
bool flag1=0, flag2=0, flag3=0;     //printing cost, paused fuel delivery on PC (once)
Ticker t1;


```

```
Start the Dispenser
```

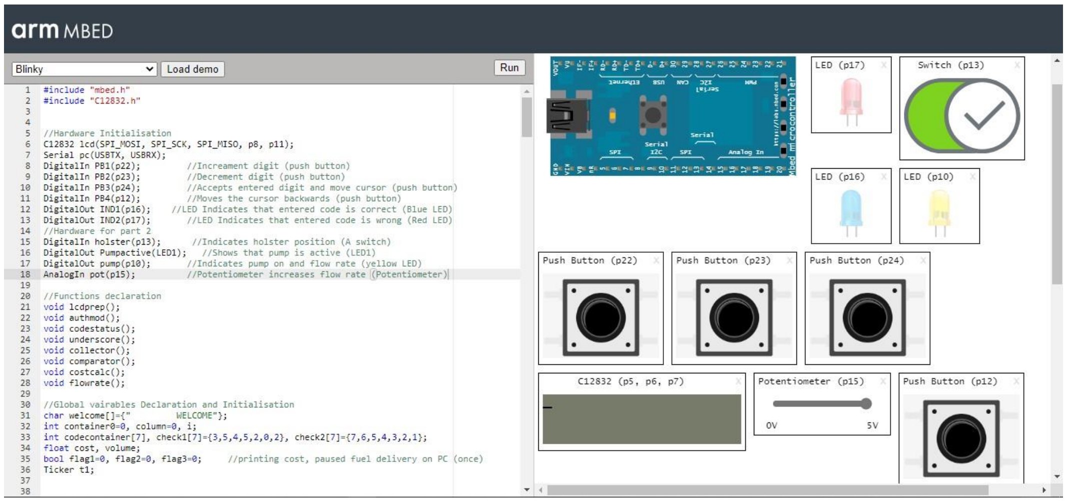
Figure 16: Inviting Attendant to lift the Nozzle from the Holster.



```
#include "mbed.h"
#include "C12832.h"


//Hardware Initialisation
C12832 lcd(SPI_MOSI, SPI_SCK, SPI_MISO, p8, p11);
Serial pc(USBTX, USBRX);
DigitalIn PB1(p22);          //Increament digit (push button)
DigitalIn PB2(p23);          //Decrement digit (push button)
DigitalIn PB3(p24);          //Accepts entered digit and move cursor (push button)
DigitalIn PB4(p12);          //Moves the cursor backwards (push button)
DigitalOut IND1(p16);    //LED Indicates that entered code is correct (Blue LED)
DigitalOut IND2(p17);        //LED Indicates that entered code is wrong (Red LED)
//Hardware for part 2
DigitalIn holster(p13);       //Indicates holster position (A switch)
DigitalOut Pumpactive(LED1);   //Shows that pump is active (LED1)
DigitalOut pump(p10);        //Indicates pump on and flow rate (yellow LED)
AnalogIn pot(p15);           //Potentiometer increases flow rate (Potentiometer)

//Functions declaration
void lcdprep();
void authmod();
void codestatus();
void underscore();
void collector();
void comparator();
void costcalc();
void flowrate();

//Global vairables Declaration and Initialisation
char welcome[]={"         WELCOME"};
int container0=0, column=0, i;
int codecontainer[7], check1[7]={3,5,4,5,2,0,2}, check2[7]={7,6,5,4,3,2,1};
float cost, volume;
bool flag1=0, flag2=0, flag3=0;     //printing cost, paused fuel delivery on PC (once)
Ticker t1;


```

```
Enter your pesonal code:
```

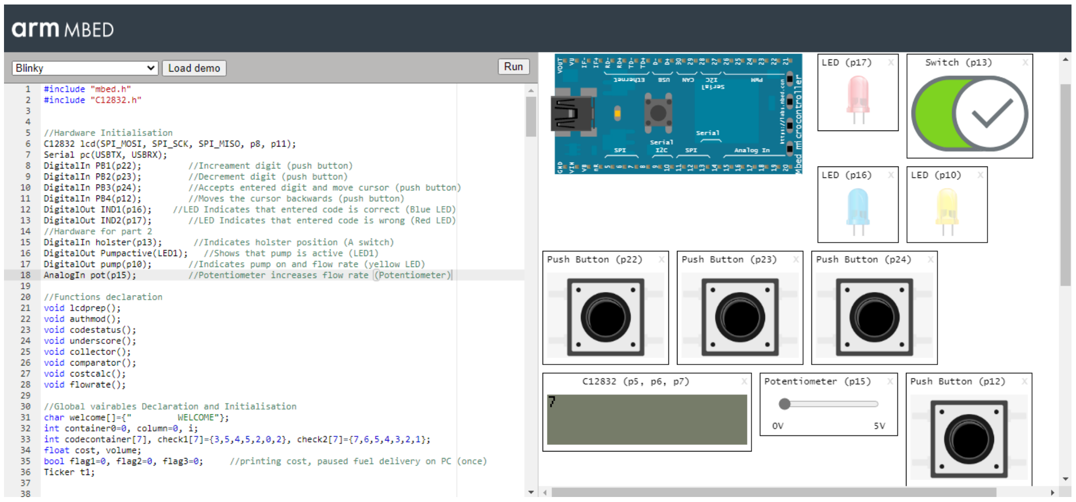
Figure 17: Inviting the Attendant to enter his/her personal code.



```
#include "mbed.h"
#include "C12832.h"


//Hardware Initialisation
C12832 lcd(SPI_MOSI, SPI_SCK, SPI_MISO, p8, p11);
Serial pc(USBTX, USBRX);
DigitalIn PB1(p22);          //Increament digit (push button)
DigitalIn PB2(p23);          //Decrement digit (push button)
DigitalIn PB3(p24);          //Accepts entered digit and move cursor (push button)
DigitalIn PB4(p12);          //Moves the cursor backwards (push button)
DigitalOut IND1(p16);    //LED Indicates that entered code is correct (Blue LED)
DigitalOut IND2(p17);        //LED Indicates that entered code is wrong (Red LED)
//Hardware for part 2
DigitalIn holster(p13);       //Indicates holster position (A switch)
DigitalOut Pumpactive(LED1);   //Shows that pump is active (LED1)
DigitalOut pump(p10);        //Indicates pump on and flow rate (yellow LED)
AnalogIn pot(p15);           //Potentiometer increases flow rate (Potentiometer)

//Functions declaration
void lcdprep();
void authmod();
void codestatus();
void underscore();
void collector();
void comparator();
void costcalc();
void flowrate();

//Global vairables Declaration and Initialisation
char welcome[]={"         WELCOME"};
int container0=0, column=0, i;
int codecontainer[7], check1[7]={3,5,4,5,2,0,2}, check2[7]={7,6,5,4,3,2,1};
float cost, volume;
bool flag1=0, flag2=0, flag3=0;     //printing cost, paused fuel delivery on PC (once)
Ticker t1;


```

```
_
```

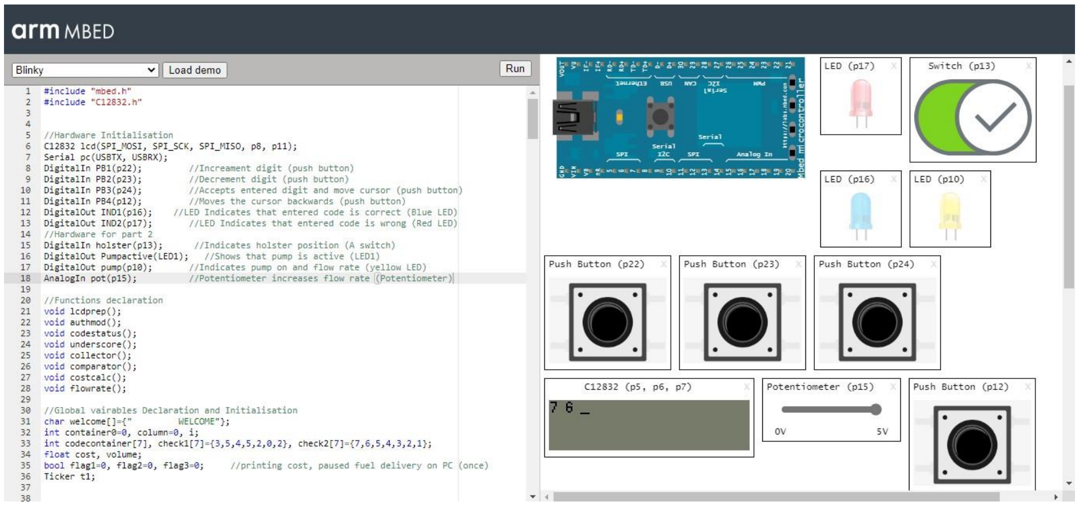
Figure 18:New screen showing "_"on the first column and row.



```
#include "mbed.h"
#include "C12832.h"


//Hardware Initialisation
C12832 lcd(SPI_MOSI, SPI_SCK, SPI_MISO, p8, p11);
Serial pc(USBTX, USBRX);
DigitalIn PB1(p22);         //Increament digit (push button)
DigitalIn PB2(p23);         //Decrement digit (push button)
DigitalIn PB3(p24);         //Accepts entered digit and move cursor (push button)
DigitalIn PB4(p12);         //Moves the cursor backwards (push button)
DigitalOut IND1(p16);    //LED Indicates that entered code is correct (Blue LED)
DigitalOut IND2(p17);       //LED Indicates that entered code is wrong (Red LED)
//Hardware for part 2
DigitalIn holster(p13);      //Indicates holster position (A switch)
DigitalOut Pumpactive(LED1);   //Shows that pump is active (LED1)
DigitalOut pump(p10);       //Indicates pump on and flow rate (yellow LED)
AnalogIn pot(p15);          //Potentiometer increases flow rate (Potentiometer)

//Functions declaration
void lcdprep();
void authmod();
void codestatus();
void underscore();
void collector();
void comparator();
void costcalc();
void flowrate();

//Global vairables Declaration and Initialisation
char welcome[]={"         WELCOME"};
int container0=0, column=0, i;
int codecontainer[7], check1[7]={3,5,4,5,2,0,2}, check2[7]={7,6,5,4,3,2,1};
float cost, volume;
bool flag1=0, flag2=0, flag3=0;     //printing cost, paused fuel delivery on PC (once)
Ticker t1;


```

```
7
```

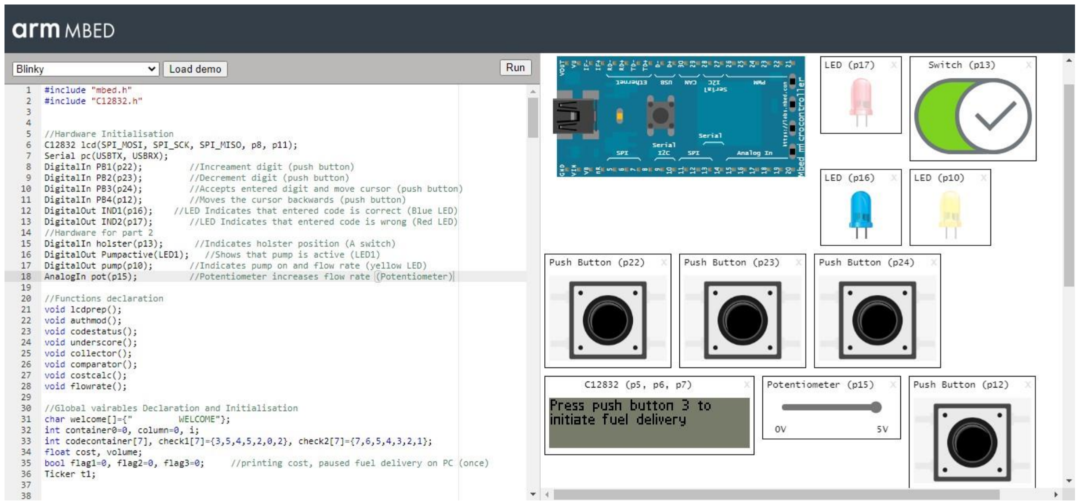
Figure 19:Entering the first digit, with pushbutton 1 to increment and pushbutton 2 to decrement.



```
#include "mbed.h"
#include "C12832.h"


//Hardware Initialisation
C12832 lcd(SPI_MOSI, SPI_SCK, SPI_MISO, p8, p11);
Serial pc(USBTX, USBRX);
DigitalIn PB1(p22);         //Increament digit (push button)
DigitalIn PB2(p23);         //Decrement digit (push button)
DigitalIn PB3(p24);         //Accepts entered digit and move cursor (push button)
DigitalIn PB4(p12);         //Moves the cursor backwards (push button)
DigitalOut IND1(p16);    //LED Indicates that entered code is correct (Blue LED)
DigitalOut IND2(p17);       //LED Indicates that entered code is wrong (Red LED)
//Hardware for part 2
DigitalIn holster(p13);      //Indicates holster position (A switch)
DigitalOut Pumpactive(LED1);   //Shows that pump is active (LED1)
DigitalOut pump(p10);       //Indicates pump on and flow rate (yellow LED)
AnalogIn pot(p15);          //Potentiometer increases flow rate (Potentiometer)

//Functions declaration
void lcdprep();
void authmod();
void codestatus();
void underscore();
void collector();
void comparator();
void costcalc();
void flowrate();

//Global vairables Declaration and Initialisation
char welcome[]={"         WELCOME"};
int container0=0, column=0, i;
int codecontainer[7], check1[7]={3,5,4,5,2,0,2}, check2[7]={7,6,5,4,3,2,1};
float cost, volume;
bool flag1=0, flag2=0, flag3=0;     //printing cost, paused fuel delivery on PC (once)
Ticker t1;


```

```
7 6 _
```

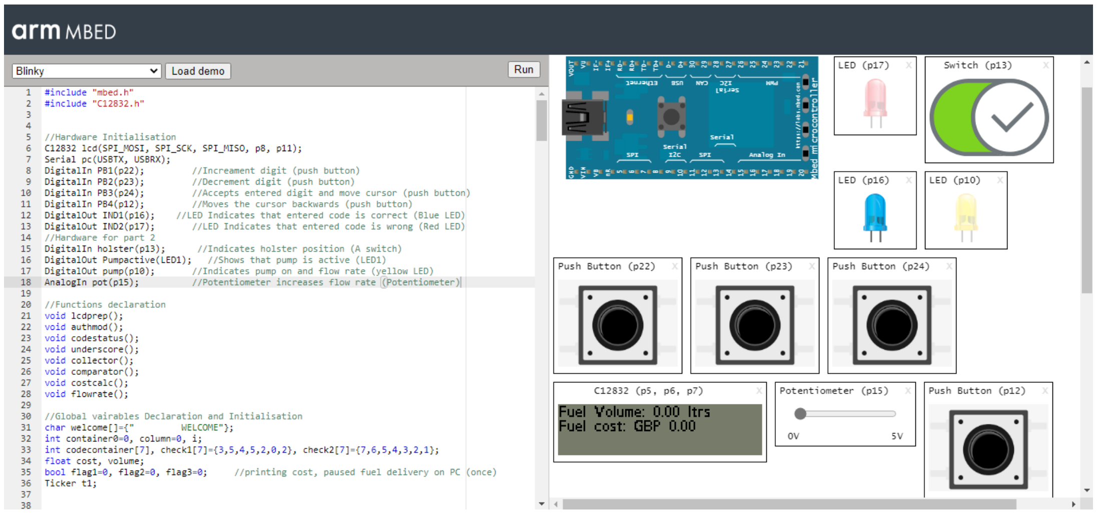
Figure 20: Accepting the entered digits using pushbutton 3 and moving the cursor to the next digit displaying "_".



```
#include "mbed.h"
#include "C12832.h"


//Hardware Initialisation
C12832 lcd(SPI_MOSI, SPI_SCK, SPI_MISO, p8, p11);
Serial pc(USBTX, USBRX);
DigitalIn PB1(p22);         //Increament digit (push button)
DigitalIn PB2(p23);         //Decrement digit (push button)
DigitalIn PB3(p24);         //Accepts entered digit and move cursor (push button)
DigitalIn PB4(p12);         //Moves the cursor backwards (push button)
DigitalOut IND1(p16);    //LED Indicates that entered code is correct (Blue LED)
DigitalOut IND2(p17);       //LED Indicates that entered code is wrong (Red LED)
//Hardware for part 2
DigitalIn holster(p13);      //Indicates holster position (A switch)
DigitalOut Pumpactive(LED1);   //Shows that pump is active (LED1)
DigitalOut pump(p10);       //Indicates pump on and flow rate (yellow LED)
AnalogIn pot(p15);          //Potentiometer increases flow rate (Potentiometer)

//Functions declaration
void lcdprep();
void authmod();
void codestatus();
void underscore();
void collector();
void comparator();
void costcalc();
void flowrate();

//Global vairables Declaration and Initialisation
char welcome[]={"         WELCOME"};
int container0=0, column=0, i;
int codecontainer[7], check1[7]={3,5,4,5,2,0,2}, check2[7]={7,6,5,4,3,2,1};
float cost, volume;
bool flag1=0, flag2=0, flag3=0;     //printing cost, paused fuel delivery on PC (once)
Ticker t1;


```

```
Press push button 3 to
initiate fuel delivery
```

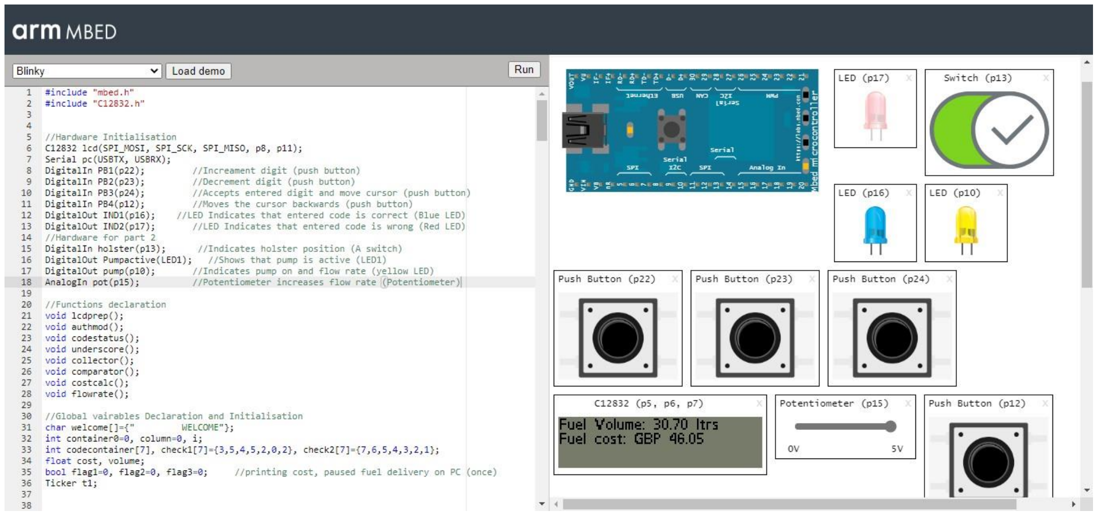
Figure 21: Pushbutton 3 submits the code and the blue LED lights up to indicate a correct code and displaying fuel delivery request on the attendant's LCD.



```
#include "mbed.h"
#include "C12832.h"


//Hardware Initialisation
C12832 lcd(SPI_MOSI, SPI_SCK, SPI_MISO, p8, p11);
Serial pc(USBTX, USBRX);
DigitalIn PB1(p22);         //Increament digit (push button)
DigitalIn PB2(p23);         //Decrement digit (push button)
DigitalIn PB3(p24);         //Accepts entered digit and move cursor (push button)
DigitalIn PB4(p12);         //Moves the cursor backwards (push button)
DigitalOut IND1(p16);    //LED Indicates that entered code is correct (Blue LED)
DigitalOut IND2(p17);       //LED Indicates that entered code is wrong (Red LED)
//Hardware for part 2
DigitalIn holster(p13);      //Indicates holster position (A switch)
DigitalOut Pumpactive(LED1);   //Shows that pump is active (LED1)
DigitalOut pump(p10);       //Indicates pump on and flow rate (yellow LED)
AnalogIn pot(p15);          //Potentiometer increases flow rate (Potentiometer)

//Functions declaration
void lcdprep();
void authmod();
void codestatus();
void underscore();
void collector();
void comparator();
void costcalc();
void flowrate();

//Global vairables Declaration and Initialisation
char welcome[]={"         WELCOME"};
int container0=0, column=0, i;
int codecontainer[7], check1[7]={3,5,4,5,2,0,2}, check2[7]={7,6,5,4,3,2,1};
float cost, volume;
bool flag1=0, flag2=0, flag3=0;     //printing cost, paused fuel delivery on PC (once)
Ticker t1;

```

```
Fuel Volume: 0.00 ltrs
Fuel cost: GBP 0.00
```

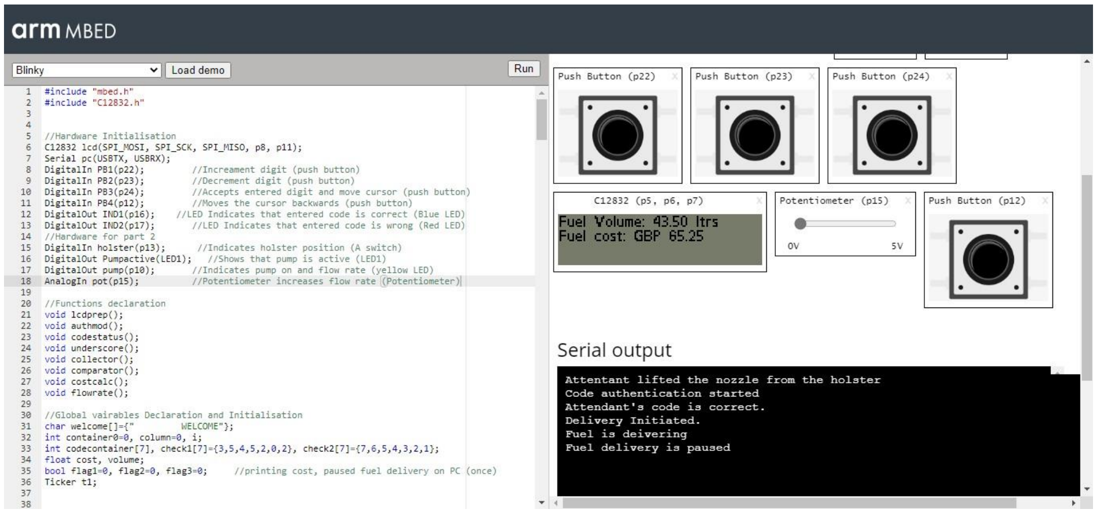
Figure 22: Pushbutton 3 pressed and delivery initiated on the LCD.



```
#include "mbed.h"
#include "C12832.h"


//Hardware Initialisation
C12832 lcd(SPI_MOSI, SPI_SCK, SPI_MISO, p8, p11);
Serial pc(USBTX, USBRX);
DigitalIn PB1(p22);         //Increament digit (push button)
DigitalIn PB2(p23);         //Decrement digit (push button)
DigitalIn PB3(p24);         //Accepts entered digit and move cursor (push button)
DigitalIn PB4(p12);         //Moves the cursor backwards (push button)
DigitalOut IND1(p16);    //LED Indicates that entered code is correct (Blue LED)
DigitalOut IND2(p17);       //LED Indicates that entered code is wrong (Red LED)
//Hardware for part 2
DigitalIn holster(p13);      //Indicates holster position (A switch)
DigitalOut Pumpactive(LED1);   //Shows that pump is active (LED1)
DigitalOut pump(p10);       //Indicates pump on and flow rate (yellow LED)
AnalogIn pot(p15);          //Potentiometer increases flow rate (Potentiometer)

//Functions declaration
void lcdprep();
void authmod();
void codestatus();
void underscore();
void collector();
void comparator();
void costcalc();
void flowrate();

//Global vairables Declaration and Initialisation
char welcome[]={"         WELCOME"};
int container0=0, column=0, i;
int codecontainer[7], check1[7]={3,5,4,5,2,0,2}, check2[7]={7,6,5,4,3,2,1};
float cost, volume;
bool flag1=0, flag2=0, flag3=0;     //printing cost, paused fuel delivery on PC (once)
Ticker t1;

```

```
Fuel Volume: 30.70 ltrs
Fuel cost: GBP 46.05
```

Figure 23: Potentiometer is turned to 5v, turning on LED1 to indicate pump activated, and blinking the yellow LED to indicate flow rate. LCD indicating accumulative fuel in litres and cost in GBP.

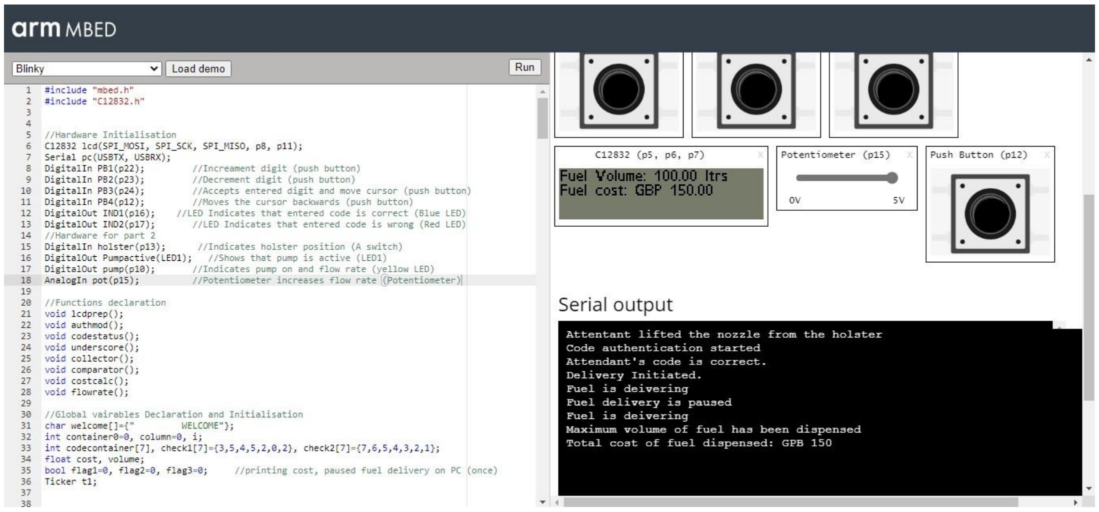


```
#include "mbed.h"
#include "C12832.h"


//Hardware Initialisation
C12832 lcd(SPI_MOSI, SPI_SCK, SPI_MISO, p8, p11);
Serial pc(USBTX, USBRX);
DigitalIn PB1(p22);         //Increament digit (push button)
DigitalIn PB2(p23);         //Decrement digit (push button)
DigitalIn PB3(p24);         //Accepts entered digit and move cursor (push button)
DigitalIn PB4(p12);         //Moves the cursor backwards (push button)
DigitalOut IND1(p16);    //LED Indicates that entered code is correct (Blue LED)
DigitalOut IND2(p17);       //LED Indicates that entered code is wrong (Red LED)
//Hardware for part 2
DigitalIn holster(p13);      //Indicates holster position (A switch)
DigitalOut Pumpactive(LED1);   //Shows that pump is active (LED1)
DigitalOut pump(p10);       //Indicates pump on and flow rate (yellow LED)
AnalogIn pot(p15);          //Potentiometer increases flow rate (Potentiometer)

//Functions declaration
void lcdprep();
void authmod();
void codestatus();
void underscore();
void collector();
void comparator();
void costcalc();
void flowrate();

//Global vairables Declaration and Initialisation
char welcome[]={"         WELCOME"};
int container0=0, column=0, i;
int codecontainer[7], check1[7]={3,5,4,5,2,0,2}, check2[7]={7,6,5,4,3,2,1};
float cost, volume;
bool flag1=0, flag2=0, flag3=0;     //printing cost, paused fuel delivery on PC (once)
Ticker t1;

```

```
Fuel Volume: 43.50 ltrs
Fuel cost: GBP 65.25
```

```
Attentant lifted the nozzle from the holster
Code authentication started
Attendant's code is correct.
Delivery Initiated.
Fuel is deivering
Fuel delivery is paused
```

Figure 24: Interrupted delivery by returning Potentiometer to 0v. Also resumes by increasing the output voltage of the potentiometer.



```cpp
#include "mbed.h"
#include "C12832.h"


//Hardware Initialisation
C12832 lcd(SPI_MOSI, SPI_SCK, SPI_MISO, p8, p11);
Serial pc(USBTX, USBRX);
DigitalIn PB1(p22);         //Increament digit (push button)
DigitalIn PB2(p23);         //Decrement digit (push button)
DigitalIn PB3(p24);         //Accepts entered digit and move cursor (push button)
DigitalIn PB4(p12);         //Moves the cursor backwards (push button)
DigitalOut IND1(p16);    //LED Indicates that entered code is correct (Blue LED)
DigitalOut IND2(p17);       //LED Indicates that entered code is wrong (Red LED)
//Hardware for part 2
DigitalIn holster(p13);      //Indicates holster position (A switch)
DigitalOut Pumpactive(LED1);   //Shows that pump is active (LED1)
DigitalOut pump(p10);       //Indicates pump on and flow rate (yellow LED)
AnalogIn pot(p15);          //Potentiometer increases flow rate (Potentiometer)

//Functions declaration
void lcdprep();
void authmod();
void codestatus();
void underscore();
void collector();
void comparator();
void costcalc();
void flowrate();

//Global vairables Declaration and Initialisation
char welcome[]={"        WELCOME"};
int container0=0, column=0, i;
int codecontainer[7], check1[7]={3,5,4,5,2,0,2}, check2[7]={7,6,5,4,3,2,1};
float cost, volume;
bool flag1=0, flag2=0, flag3=0;     //printing cost, paused fuel delivery on PC (once)
Ticker t1;
```

```
Fuel Volume: 100.00 ltrs
Fuel cost: GBP 150.00
```

Serial output

```
Attentant lifted the nozzle from the holster
Code authentication started
Attendant's code is correct.
Delivery Initiated.
Fuel is deivering
Fuel delivery is paused
Fuel is deivering
Maximum volume of fuel has been dispensed
Total cost of fuel dispensed: GPB 150
```

Figure 25: Total cost is transmitted to the Cashier’s computer screen. Also, prompts for all events occurring in the Dispenser shows up on the Cashier’s Computer screen, including every time the delivery is paused or resumed.

6. REFERENCES

Arm Limited. (2021a). *Architecting a Smarter World – Arm.* Arm | The Architecture for the Digital World. https://www.arm.com/

Arm Limited. (2021b). *Handbook | Mbed.* arm MBED. https://os.mbed.com/handbook/Homepage

Arm Limited. (2021c). *NUCLEO-F103RB | Mbed.* arm MBED. https://os.mbed.com/platforms/ST-Nucleo-F103RB/

Hafeez, M. (2021). NME3545-2021 Assignment Brief (simulator) 2021: Fuel Delivery System. . https://brightspace.hud.ac.uk/d2l/le/content/131529/viewContent/956088/View

Ibrahim, D. (2019). *ARM-based microcontroller projects using mbed* (First; 1 ed.). Newnes.

IEEE. (2021). *IEEE 830-1998 - IEEE Recommended Practice for Software Requirements Specifications.* IEEE Standards Association. https://standards.ieee.org/standard/830-1998.html

Iniewski, K. (2012). *Embedded systems: hardware, design, and implementation* (1. Aufl.; 1 ed.). John Wiley & Sons, Inc. 10.1002/9781118468654

Johnson, A., & Hafeez, M. (2021). NME3545-2021 Lecture_1: Embedded Systems. https://brightspace.hud.ac.uk/d2l/le/content/131529/viewContent/915399/View

Toulson, R. (2017). *Fast and effective embedded systems design: applying the arm mbed / ‡c Rob Toulson, Tim Wilmshurst* (Second ed.). Elsevier, ‡c 2017.

Toulson, R., & Wilmshurst, T. (2016). *Fast and effective embedded systems design: applying the ARM mbed*. Newnes.

Vystavěl, R. (2017). *C# Programming for Absolute Beginners* (1st 2017.; 1 ed.). Apress.

7. APPENDIX

```
/*
Description: This is the program for a Fuel Delivery system, intended to be
used on an ARM Nucleo STM32F103 mirco-controller.
*/

#include "mbed.h"
#include "C12832.h"


//Hardware Initialisation
C12832 lcd(SPI_MOSI, SPI_SCK, SPI_MISO, p8, p11);
Serial pc(USBTX, USBRX);
DigitalIn PB1(p22);         //Increment digit (push button)
DigitalIn PB2(p23);         //Decrement digit (push button)
DigitalIn PB3(p24);         //Accepts entered digit and move cursor (push button)
DigitalIn PB4(p12);         //Moves the cursor backwards (push button)
DigitalOut IND1(p16);       //LED Indicates that entered code is correct (Blue LED)
DigitalOut IND2(p17);       //LED Indicates that entered code is wrong (Red LED)
DigitalIn holster(p13);     //Indicates holster position (A switch)
DigitalOut Pumpactive(LED1);   //Shows that pump is active (LED1)
DigitalOut pump(p10);       //Indicates pump on and flow rate (yellow LED)
AnalogIn pot(p15);          //Potentiometer increases flow rate (Potentiometer)

//Functions declaration
void lcdprep();
void authmod();
void codestatus();
void underscore();
void collector();
void comparator();
void costcalc();
void flowrate();

//Global vairables Declaration and Initialisation
char welcome[]={"Now selling at 10% discount!"};
int container0=0, column=0, i;
int codecontainer[7], check1[7]={3,5,4,5,2,0,2}, check2[7]={7,6,5,4,3,2,1};
float cost, volume;
bool flag1=0, flag2=0, flag3=0;     //printing cost, paused fuel delivery on PC (once)
Ticker t1;



//Main function
int main()
{
```

```
while(IND1==0 && holster==0)
{
   for(int i=0; i<35; i++)
   {
      lcd.printf("");
      lcd._putc(welcome[i]);
      wait(0.1);
   }
   wait(1);
   while(IND1==0)
   {
      lcdprep();
      lcd.printf("Start the Dispenser");
      if(holster==1)
      {
         authmod();
         wait(0.2);
      }
      else
      {
         IND2.write(false);
         wait(0.2);
      }
   }
   if(IND1==1)
   {
      printf("Attendant's code is correct.\n");
      lcdprep();
      lcd.printf("Press push button 3 to \ninitiate fuel delivery");
   }
   while(holster==1)
   {
      if(PB3==1)
      {
         lcdprep();
         lcd.printf("Fuel Volume: %.2f ltrs\nFuel cost: GBP %.2f",volume,cost);
         printf("Delivery Initiated.\n");
         flowrate();
      }
      if(cost<150 && holster==0)  //allows metering to work if code starts afresh
      {
         IND1=0;
         printf("Pump has been turned off\n");
         printf("Total cost of fuel dispensed: GPB %.0f\n", cost);
         wait(1);
         volume=0;
         cost=0;
         flag1=0;
         flag2=0;
```

```
                t1.detach();
                Pumpactive.write(0);
                container0=0;   //ensures codecollector doesn't run until PB3 is pressed
                lcdprep();
            }
            //keeps cost on pc screen
            else if(cost>=150 && holster==1 && flag3==0)
            {
                IND1=0;
                printf("Maximum volume of fuel has been dispensed\n");
                printf("Total cost of fuel dispensed: GPB %.0f\n", cost);
                flag3=1;        //makes cost display once on Cashier's monitor
            }
            wait_ms(2);
        }
        if(cost>=150 && holster==0)  //allows metering to work if code starts afresh
        {
            IND1=0;
            printf("Attendant returned the nozzle to the holster\n");
            wait(1);
            volume=0;
            cost=0;
            flag1=0;
            flag2=0;
            flag3=0;
            t1.detach();
            Pumpactive.write(0);
            container0=0;
            lcdprep();
        }
        else if(holster==0)
        {
            IND1=0;
            t1.detach();
            Pumpactive.write(0);
            container0=0;        //ensures codecollector doesn't run until PB3 is pressed
            printf("Attendant returned the nozzle to the holster\n");
            lcdprep();
        }
        wait(1);
    }
}



/*
Function definitions starts here
*/
void lcdprep()
```

```
{
   lcd.cls();
   lcd.locate(0,0);
}

void authmod()
{
   codestatus();
   underscore();
   collector();
}

void codestatus()
{
   for(int clr=0; clr<7; clr++)    //Clears the array
   {
      codecontainer[clr]=-1;
   }
   if(IND2==1)             //When passcode was wrong in the first try
   {
      lcdprep();
      lcd.printf("Renter the code: \n");
   }
   else                    //First try
   {
      printf("Attentant lifted the nozzle from the holster\n");
      lcdprep();
      lcd.printf("Enter your pesonal code: \n");
   }
}

void underscore()
{
   while(container0==0 && holster==1)
   {
      if(PB3==1)
      {
         lcdprep();
         lcd.printf("_");
         container0=1;
         printf("Code authentication started\n");
      }
      else
      {
         container0=0;
      }
      wait(0.2);
   }
}
```

```
void collector()
{
   if(IND1==0 && holster==1)
   {
      for(i=0; i<7; i++)
      {
         if(holster==1)
         {
            lcd.locate(column,0);
            lcd.printf("_");
         }
         while(PB3==0 && holster==1)
         {
            if(PB1==1 && codecontainer[i]<9)
            {
               codecontainer[i]++;
               lcd.locate(column,0);
               lcd.printf("%d", codecontainer[i]);
            }
            else if(PB2==1 && codecontainer[i]>0)
            {
               codecontainer[i]--;
               lcd.locate(column,0);
               lcd.printf("%d", codecontainer[i]);
            }
            else if(PB4==1 && column>0)
            {
               codecontainer[i]=-1;
               lcd.locate(column,0);
               lcd.printf(" ");
               i--;
               column=column-10;
               lcd.locate(column,0);
               codecontainer[i]=-1;
               lcd.printf("_");
            }
            wait(0.2);
         }
         column=column+10;
         wait(0.2);
      }
      comparator();
   }
   else if(IND1==0 && holster==0)
   {
      container0=0;
      printf("Attendant returned the nozzle to the holster\n");
      lcdprep();
```

```
    }
}

void comparator()
{
    for(int j=0; j<7; j++)
    {
        if(holster==1 && codecontainer[j]==check1[j] || codecontainer[j]==check2[j])
        {
            IND1=1;
            IND2=0;
            column=0;
        }
        else if(holster==1 && codecontainer[j]!=check1[j] || codecontainer[j]!=check2[j])
        {
            IND1=0;
            IND2=1;
            lcd.cls();
            column=0;
            container0=0;
            printf("Attendant's code is wrong.\n");
            break;
        }
        else if(holster==0)
        {
            IND1=0;
            IND2=0;
            lcd.cls();
            column=0;
            container0=0;
            printf("Attendant returned the nozzle to the holster\n");
            break;
        }
    }
}


void costcalc()
{
    if(cost<=150 && holster==1)
    {
        pump = !pump;
        volume=volume+0.1;
        cost= volume*1.5;
    }
    else
    {
        pump.write(0);
    }
    if(pot>0 && holster==1 && cost<=150 && flag1==0)
    {
        printf("Fuel is deivering\n");
        flag1=1;
    }
    else if(pot==0 && holster==1 && cost<=150 && flag2==0)
    {
        printf("Fuel delivery is paused\n");
        flag2=1;
    }
}

void flowrate(){
    while (IND1==1 && holster==1 && cost<=150)
    {
        if(pot*5>0 && pot*5<=1)
        {
            t1.attach(callback(&costcalc), 1.0f);
            lcdprep();
            lcd.printf("Fuel Volume: %.2f ltrs\nFuel cost: GBP %.2f",volume,cost);
            Pumpactive.write(1);
            flag2=0;                //sets bool for when delivery is paused
            wait(1.0);
        }
        else if(pot*5>1 && pot*5<=2)
        {
            t1.attach(callback(&costcalc), 0.8f);
            lcdprep();
            lcd.printf("Fuel Volume: %.2f ltrs\nFuel cost: GBP %.2f",volume,cost);
            Pumpactive.write(1);
            flag2=0;
            wait(0.8);
        }
        else if(pot*5>2 && pot*5<=3)
        {
            t1.attach(callback(&costcalc), 0.6f);
            lcdprep();
            lcd.printf("Fuel Volume: %.2f ltrs\nFuel cost: GBP %.2f",volume,cost);
            Pumpactive.write(1);
            flag2=0;
            wait(0.6);
        }
        else if(pot*5>3 && pot*5<=4)
        {
            t1.attach(callback(&costcalc), 0.3f);
            lcdprep();
            lcd.printf("Fuel Volume: %.2f ltrs\nFuel cost: GBP %.2f",volume,cost);
            Pumpactive.write(1);
            flag2=0;
            wait(0.3);
        }
        else if(pot*5>4 && pot*5<5)
        {
            t1.attach(callback(&costcalc), 0.1f);
            lcdprep();
            lcd.printf("Fuel Volume: %.2f ltrs\nFuel cost: GBP %.2f",volume,cost);
            Pumpactive.write(1);
            flag2=0;
            wait(0.1);
        }
        else if(pot*5>4 && pot*5<=5)
        {
            t1.attach(callback(&costcalc), 0.065f);
            lcdprep();
            lcd.printf("Fuel Volume: %.2f ltrs\nFuel cost: GBP %.2f",volume,cost);
            Pumpactive.write(1);
            flag2=0;            //sets bool for when delivery is paused
            wait(0.065);
        }
        else
        {
            t1.detach();
            Pumpactive.write(0);
            pump.write(0);
            flag1=0;            //sets bool for when fuel is delivering
            wait_us(0.2);
        }
    }
}
```