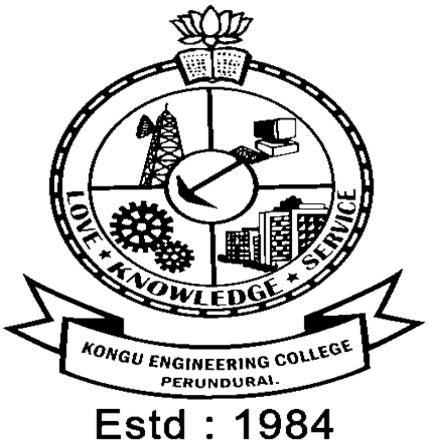
KONGU ENGINEERING COLLEGE

(Autonomous)

Perundurai, Erode– 638060.



LABORATORY RECORD

Name : _____

Register Number: _____

Course code : _____

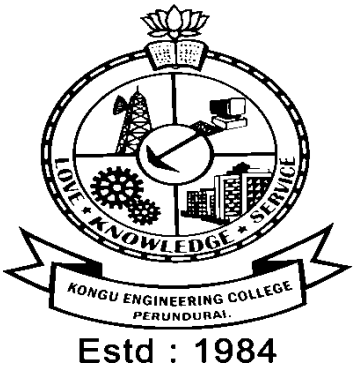
Course name : _____

Semester & Branch:_____

KONGU
Assuring the Best

KONGU ENGINEERING COLLEGE

(Autonomous)
Perundurai, Erode–638060.



Department of Electronics and Communication Engineering

22ECC51– Embedded Systems and IoT Laboratory Record

Name_____Programme_____

Branch_____Section_____Semester_____

Register Number _____

Certified that this is a bonafide record of work done by the above student of the ***22ECC51– Embedded Systems and IoT Laboratory Record*** *during the year* ***2024–2025.***

Signature of Lab in-charge **Signature of the HOD**

Submitted for the practical examination held on _____

Examiner-I **Examiner-II**

INDEX

S. No.	Date	Name of the Experiment	Submission date	Page No.	Marks awarded	Signature
1.		Device ON / OFF using PIC 16F877A microcontroller (Relay and LED)				
2.		Interfacing of LCD with PIC 16F877A microcontroller				
3.		Analog sensor interfacing with PIC16F877A microcontroller				
4.		Design of clock using Real Time Clock with PIC 16F877A microcontroller				
5.		Control LEDs via Classic Bluetooth using the WDM board. (Bluetooth Communication)				
6.		Push sensor data to a cloud platform and create a dashboard via the HTTP protocol using the WDM board. (Wifi Communication)				
7.		Integrate a third-party application server for data storage and monitoring. (LoRaWAN Communication)				
8.		Integrate a third-party application server for controlling the motor using the WDM board. (LoRaWAN Communication)				
Average Marks						

Exp. No: 1 Date: 05.08.2024	Device ON / OFF using PIC 16F877A microcontroller (Relay and LED)

Aim:

To design and develop an embedded C program to turn ON / OFF LED using PIC16F877A microcontroller and a relay.

Software Required:

1. Code Composer Studio (CCS) compiler to compile and create hex file.
2. Proteus 8.xx is used for the simulation of circuits.
3. PICkit 2 programmer for burning the hex file to PIC microcontroller.

Hardware required:

Sl. No	Apparatus name	Range	Quantity
1.	Bread board	-	1
2.	Regulated Power Supply	5V, 2A	1
3.	PIC16F877A	40-Pin PDIP	1
4.	Crystal oscillator	4 MHz	1
5.	PIC programmer and USB cable	-	1
6.	Light Emitting Diode (LED)	-	1
7.	Switch	SPST / SPDT	1
8.	Relay	12V	1
9.	BC547 - NPN Transistor	-	1
10.	Diode	1N4001	1
11.	Resistor	330 Ω, 1 k	Each 2
12.	Connecting wires	Single strand	As required

Procedure:

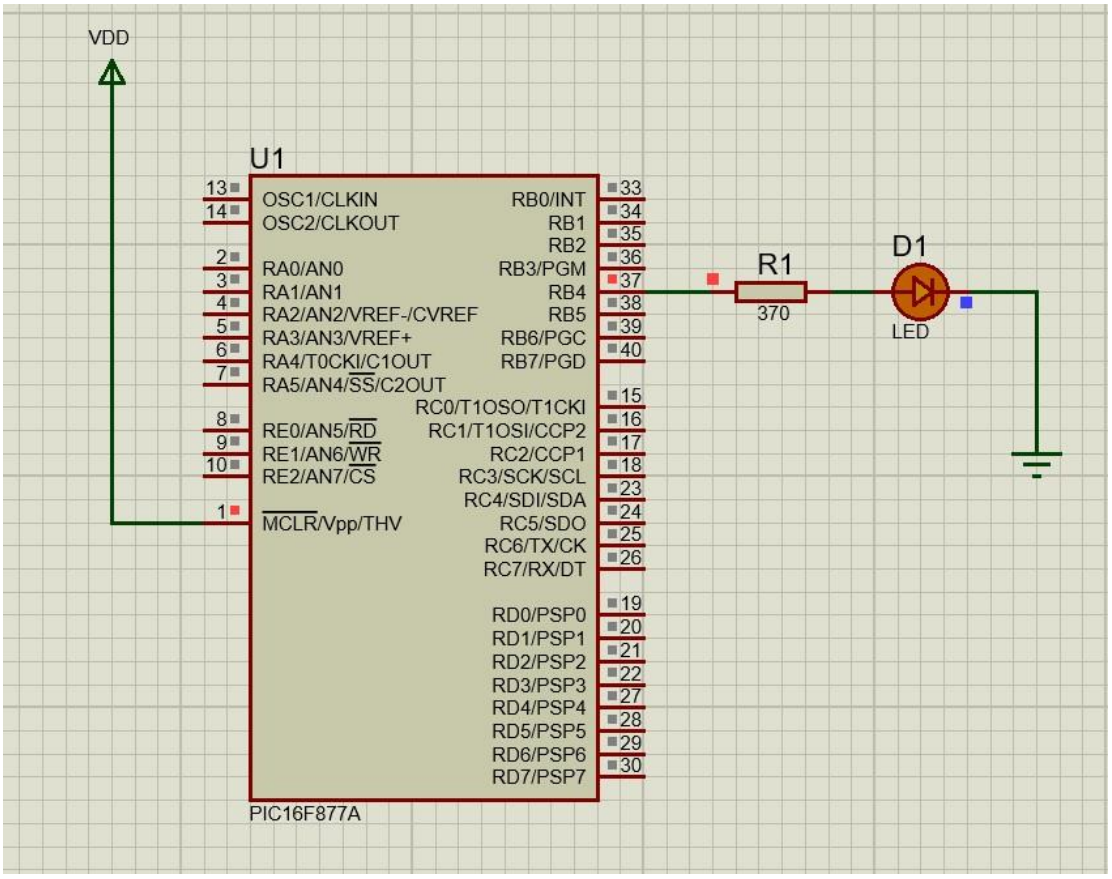
- Open PIC C compiler
- Click New File and Click Project Wizard
- Select PIC16 family, PIC16F877A.
- Select I/O ports and set the port as input or output.
- Open proteus software and load the hex. file generated form the CCS compiler.
- Run and View the Output.

A) Blinking of LED using PIC 16F877A Microcontroller

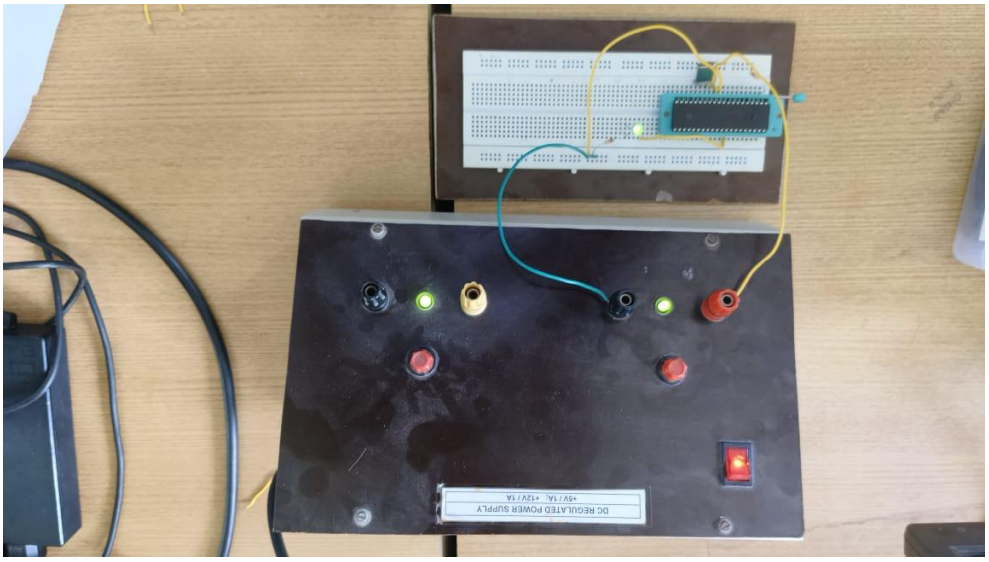
Program:

```
#include <16F877A.h>
#device ADC=16

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES NOBROWNOUT       //No brownout reset
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for
I/O

#use delay(crystal=4MHz)
#use FIXED_IO( B_outputs=PIN_B4 )

void main()
{
  while(TRUE)
  {
    //TODO: User Code
    output_high(PIN_B4);
    delay_ms(1000);
    output_low(PIN_B4);
    delay_ms(1000);
  }
}
```

Simulation output:



Hardware Output:

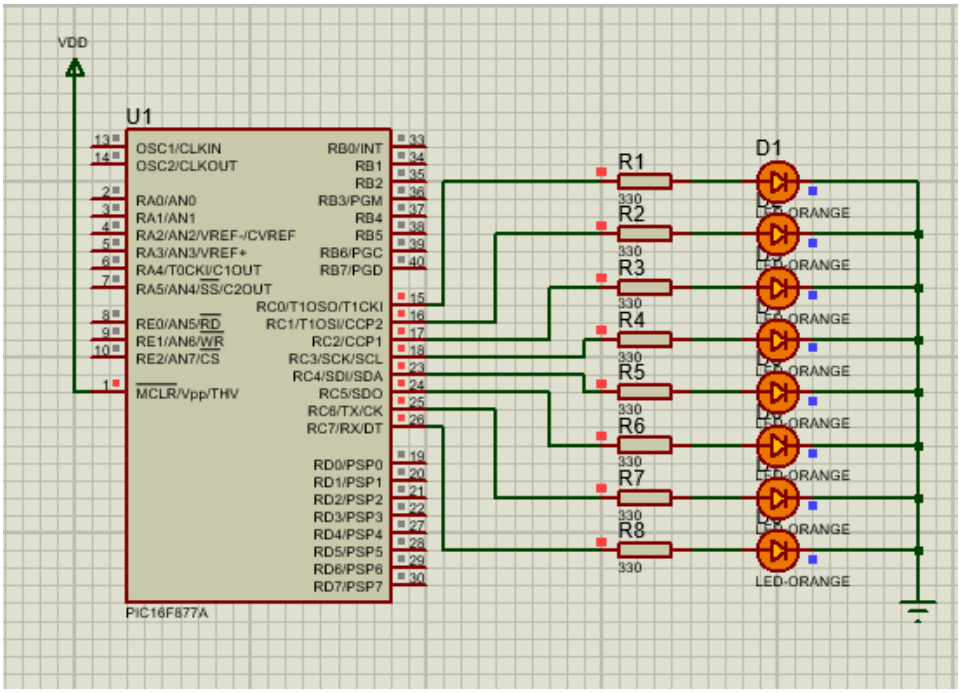


B) Blinking of 8 LED using PIC 16F877A Microcontroller

Program:

```
#include <16F877A.h>
#device ADC=16
#FUSES NOWDT
#FUSES NOBROWNOUT
#FUSES NOLVP
#use delay(crystal=4MHz)
#use FIXED_IO(
C_outputs=PIN_C7,PIN_C6,PIN_C5,PIN_C4,PIN_C3,PIN_C2,PIN_C1,PIN_C0 )


void main()
{
      while(TRUE)
     {

         output_C(0x00);
         delay_ms(1000);
         output_C(0xFF);
         delay_ms(1000);

     }
}
```

Simulation output:



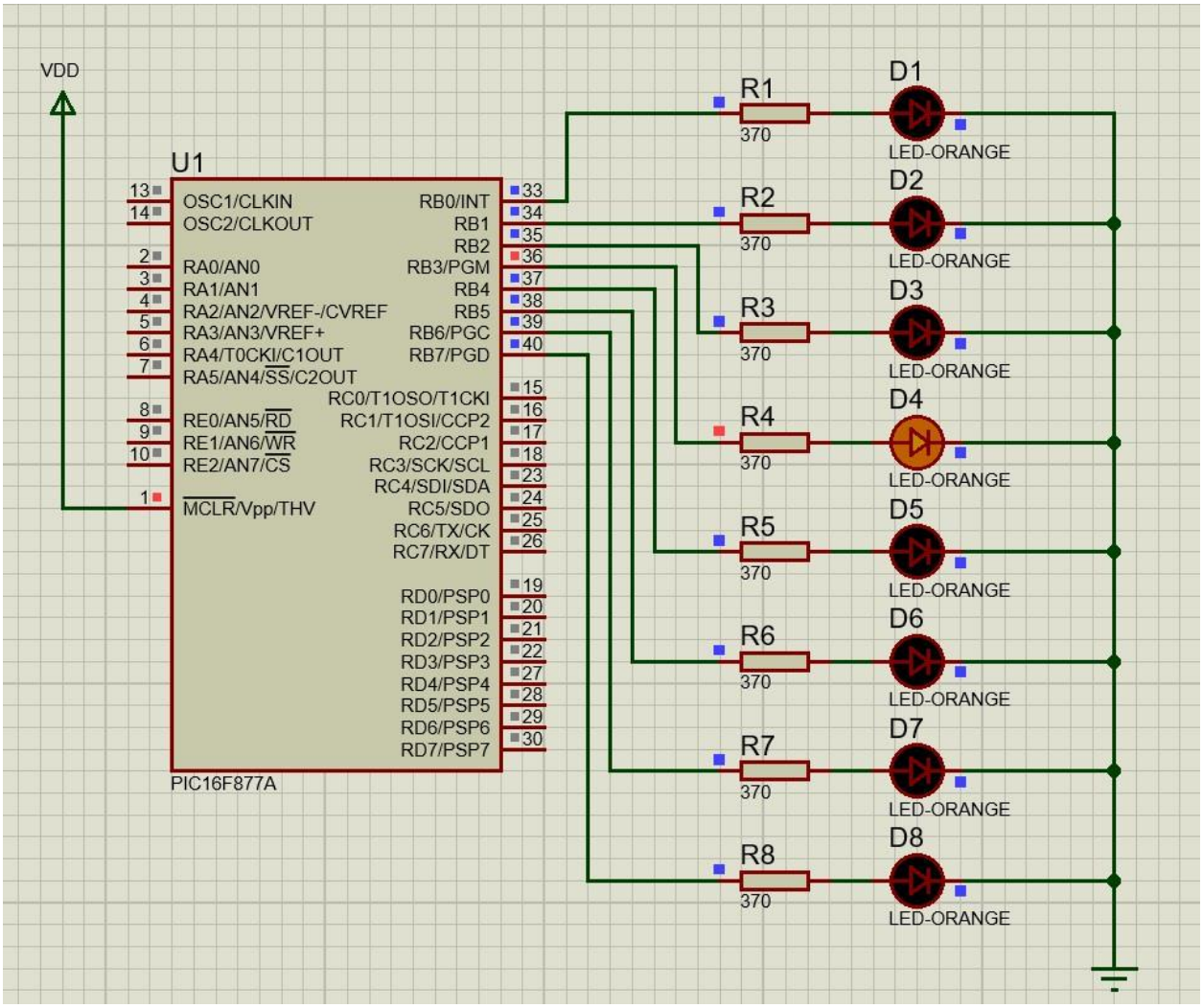
C) Scrolling of LED using PIC 16F877A Microcontroller

Program:

```
#include <16F877A.h> #device ADC=16 #FUSES NOWDT
#FUSES NOBROWNOUT #FUSES NOLVP
#use delay(crystal=4MHz)
#use FIXED_IO(
C_outputs=PIN_C7,PIN_C6,PIN_C5,PIN_C4,PIN_C3,PIN_C2,PIN_C1,PIN_C0 )
void main()
{
    while(TRUE)
    {
```

```
            output_C(0x00);
            delay_ms(500);
            output_C(0x01);
            delay_ms(500);
            output_C(0x02);
            delay_ms(500);
            output_C(0x04);
            delay_ms(500);
            output_C(0x08);
            delay_ms(500);
            output_C(0x10);
            delay_ms(500);
            output_C(0x20);
            delay_ms(500);
            output_C(0x40);
            delay_ms(500);
          output_C(0x80);
          delay_ms(500);
    }
}
```

Simulation output:

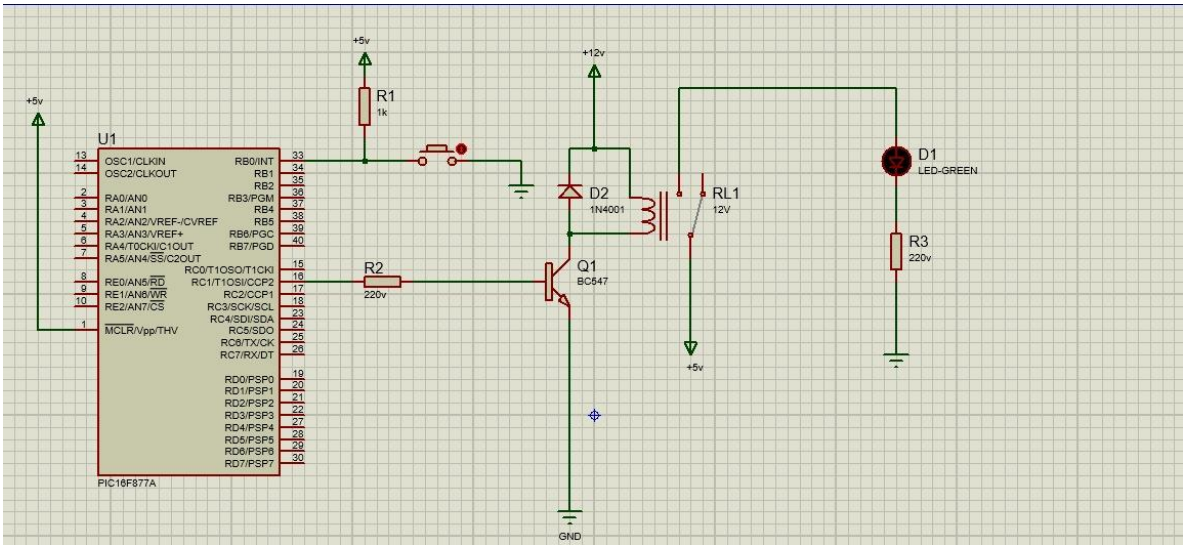


D) Switch with Relay

Program:

```
#include <16F877A.h>
#device ADC=16
#FUSES NOWDT                    //No Watch Dog Timer
#FUSES NOBROWNOUT               //No brownout reset
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for
I/O
#use delay(crystal=4MHz)

void main()
{
  while(TRUE)
  {
    if (!input(PIN_B0))
        output_high(PIN_B7);
    else
        output_low(PIN_B7);
  }
}
```

Simulation output:



Hardware Output:

Video References:

Switch Led

Relay Led

Rubrics		Marks
Conduct of Experiment (20)	Analyse the problem and develop programming constructs (15)	
	Completeness of the experiment (5)	
Observation/ Record (30)	Interpretation of the findings (15)	
	Simulation and Hardware (5)	
	Adherence to record submission deadline (5)	
	Presentation and completion of record (5)	
Viva (10)	Ability to recall the theoretical concepts	
Total (60)		

Result:

Thus the Device ON / OFF using PIC 16F877A microcontroller (Relay and LED) was is done successfully by using PIC C Compiler and Proteus Software.

Exp. No: 2 Date: 12-08-2024	**Interfacing of LCD with PIC 16F877A microcontroller**

Aim:

To design and develop an embedded C program to interface the LCD with the PIC 16F877A microcontroller.

Software Required:

1. Code Composer Studio (CCS) compiler to compile and create hex file.
2. Proteus 8.xx is used for the simulation of circuits.
3. PICkit 2 programmer for burning the hex file to PIC microcontroller.

Hardware required:

S. No	Apparatus name	Range	Quantity
1.	Bread board	-	1
2.	Regulated Power Supply	5V, 2A	1
3.	PIC16F877A	40-Pin PDIP	1
4.	Crystal oscillator	4MHz	1
5.	PIC programmer and USB cable	-	1
6.	LCD Display	16X2	1
7.	Resistor	1K	1
8.	Connecting wires	Single strand	As required

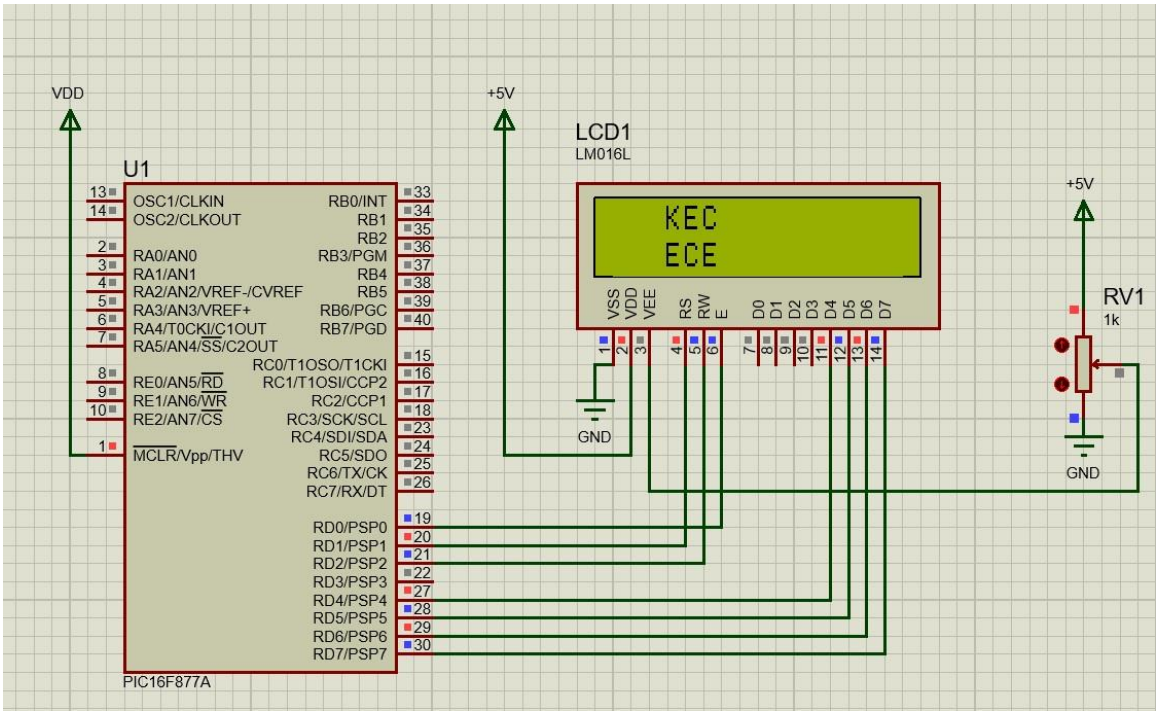
Procedure:

- Open PIC C compiler
- Click New File and Click Project Wizard
- Select PIC16 family, PIC16F877A.
- Select I/O ports and set the port as input or output.
- Open proteus software and load the hex. file generated form the CCS compiler.
- Run and View the Output.

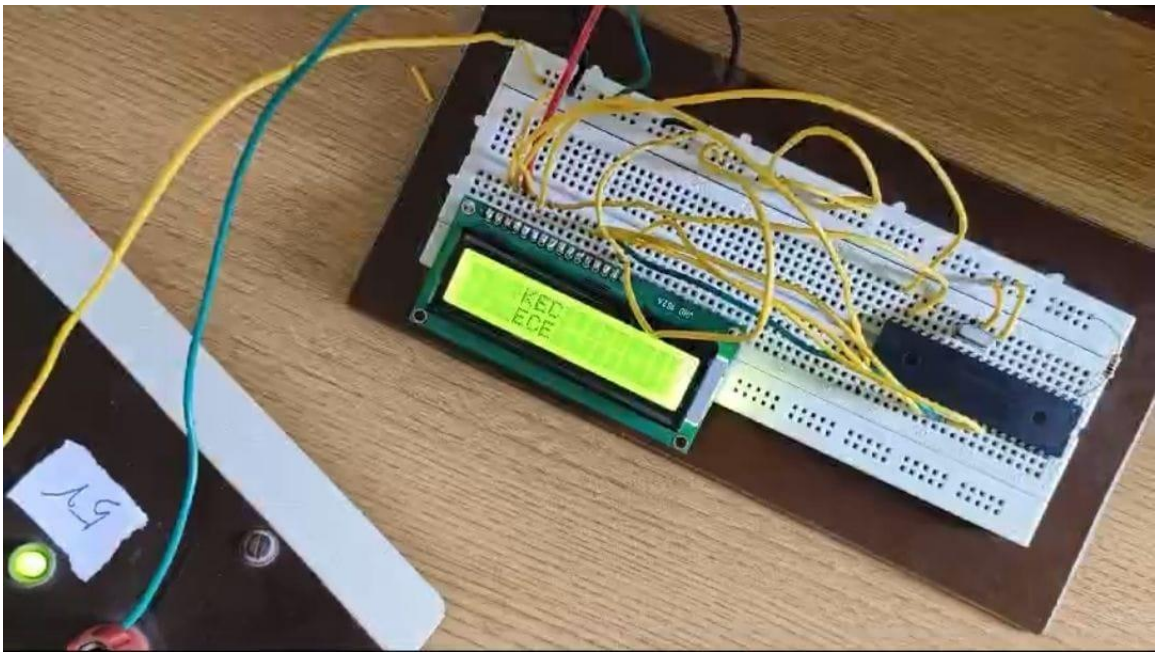
Program:

```
#include <16F877A.h>
#device ADC=16

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES PUT                      //Power Up Timer
#FUSES NOBROWNOUT               //No brownout reset
#FUSES NOLVP                    //No low voltage programming, B3(PIC16) or B5(PIC18) used for I/O

#use delay(crystal=4MHz)

#define LCD_ENABLE_PIN PIN_D0
#define LCD_RS_PIN PIN_D1
#define LCD_RW_PIN PIN_D2
#define LCD_DATA4 PIN_D4
#define LCD_DATA5 PIN_D5
#define LCD_DATA6 PIN_D6
#define LCD_DATA7 PIN_D7

#include <lcd.c>

void main()
{
  lcd_init();

  while(TRUE)
  {
    lcd_gotoxy(4,1);
    lcd_putc("KEC");
    lcd_gotoxy(4,2);
    lcd_putc("ECE");
    delay_ms(800);
  }

}
```

Simulation Output:



Hardware Output:



Video References:

Rubrics		Marks
Conduct of Experiment (20)	Analyse the problem and develop programming constructs (15)	
	Completeness of the experiment (5)	
Observation/Record (30)	Interpretation of the findings (15)	
	Simulation and Hardware (5)	
	Adherence to record submission deadline (5)	
	Presentation and completion of record (5)	
Viva (10)	Ability to recall the theoretical concepts	
Total(60)		

Result:

The design and implementation LCD with PIC 16F877A was completed and done successfully.

Exp. No : 3	Analog Sensor Interfacing with PIC 16F877A microcontroller (ADC)
Date : 19.08.2024	

Aim:

To design and develop an embedded C program to interface analog sensor with the PIC 16F877A microcontroller.

Software Required:

1. Code Composer Studio (CCS) compiler to compile and create hex file.
2. Proteus 8.13 is used for the simulation of circuits.
3. PICkit 2 programmer for burning the hex file to PIC microcontroller.

Hardware required:

S. No	Apparatus name	Range	Quantity
1.	Bread board	-	1
2.	Regulated Power Supply	5V, 2A	1
3.	PIC16F877A	40-Pin PDIP	1
4.	Crystal oscillator	4MHz	1
5.	PIC programmer and USB cable	-	1
6.	Analog Sensor	-	1
7.	LCD Display	16X2	1
8.	Potentiometer	10K	1
9.	Resistor	1K	1
10.	Connecting wires	Single strand	As required

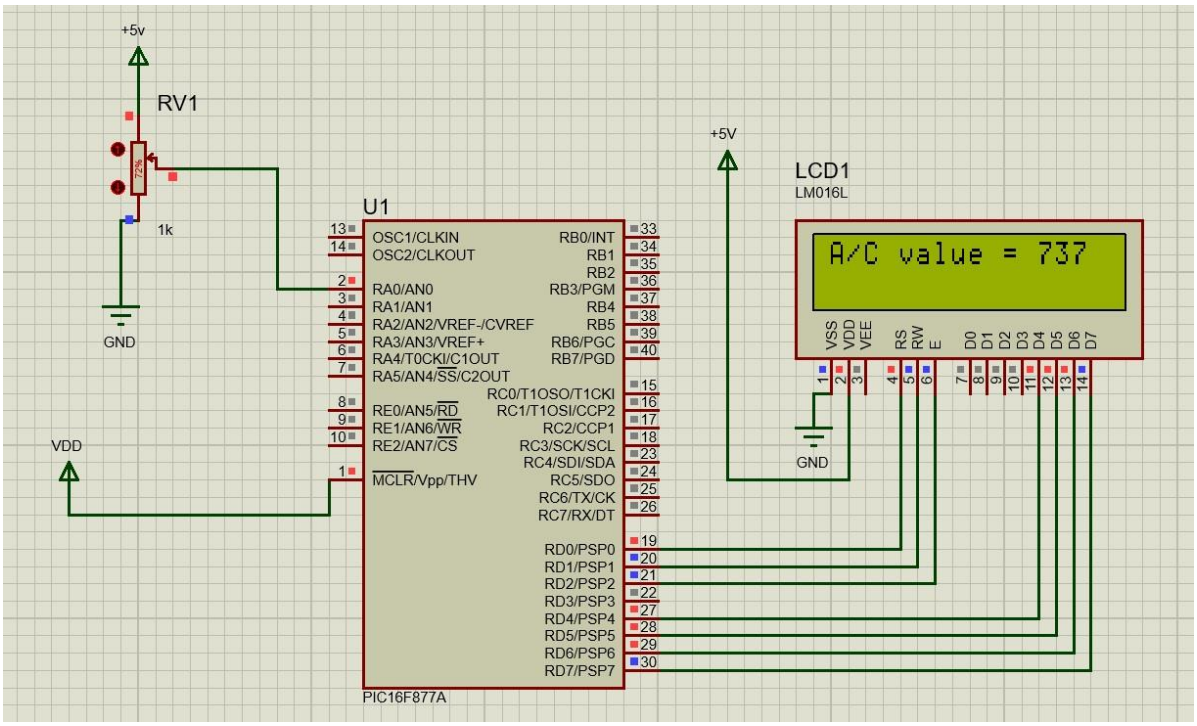
Procedure:

- Open PIC C compiler
- Click New File and Click Project Wizard
- Select PIC16 family, PIC16F877A.
- Select I/O ports and set the port as input or output.
- Open proteus software and load the hex. file generated form the CCS compiler.
- Run and View the Output.

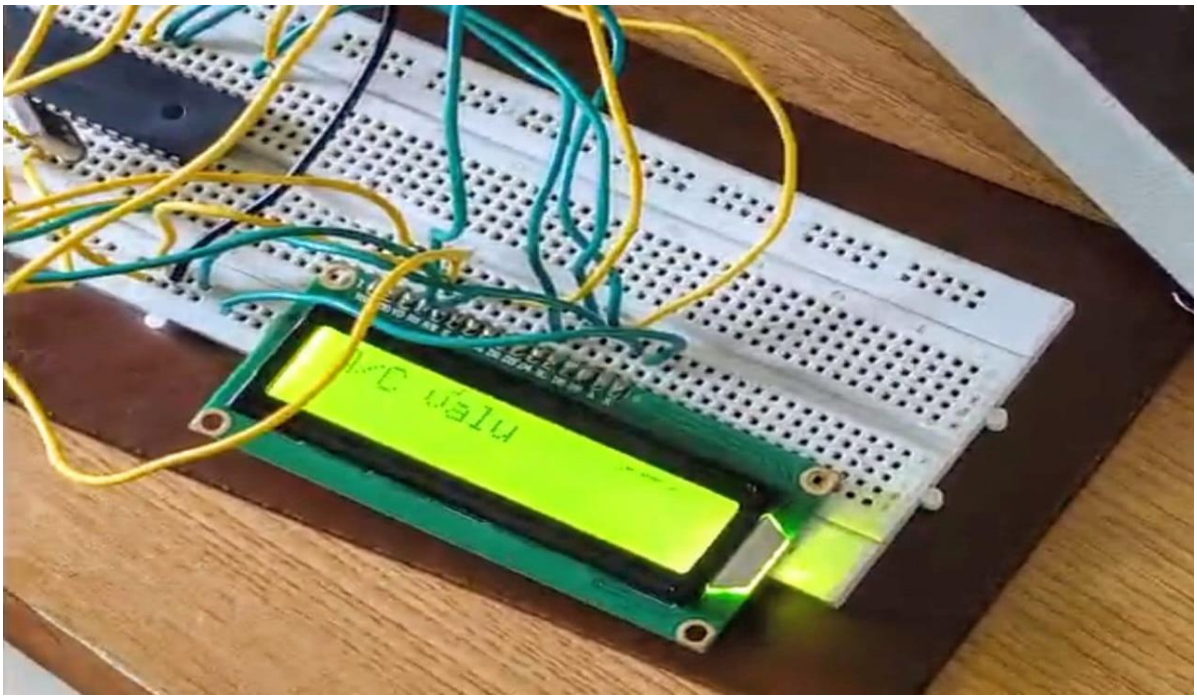
Program:

```
#include <16F877A.h>
#device ADC=10
#FUSES NOWDT                    //No Watch Dog Timer
#FUSES PUT                      //Power Up Timer
#FUSES NOBROWNOUT               //No brownout reset
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#use delay(crystal=4MHz)

#define LCD_ENABLE_PIN PIN_D2
#define LCD_RS_PIN PIN_D0
#define LCD_RW_PIN PIN_D1
#define LCD_DATA4 PIN_D4
#define LCD_DATA5 PIN_D5
#define LCD_DATA6 PIN_D6
#define LCD_DATA7 PIN_D7

#include <lcd.c>

void main()

{

   setup_adc_ports(AN0_AN1_AN3);
   setup_adc(ADC_CLOCK_INTERNAL);
   lcd_init();

   while(TRUE)
   {
      int16 value;
      set_adc_channel(0);
      read_adc(ADC_START_ONLY);   // SOC
      int1 done = adc_done();
      while(!done)                //EOC
      {
         done = adc_done();
      }
      value = read_adc(ADC_READ_ONLY);
      printf(lcd_putc,"\fA/C value = %ld", value);
      delay_ms(500);
   }
 }
```

Simulation Output:



Hardware Output:



Video References:

Rubrics		Marks
Conduct of Experiment (20)	Analyse the problem and develop programming constructs (15)	
	Completeness of the experiment (5)	
Observation/Record (30)	Interpretation of the findings (15)	
	Simulation and Hardware (5)	
	Adherence to record submission deadline (5)	
	Presentation and completion of record (5)	
Viva (10)	Ability to recall the theoretical concepts	
Total(60)		

Result:

The design and implementation of Analog Sensor with PIC 16F877A microcontroller was done and verified with LCD display.

Exp. No: 4	Design of clock using Real Time Clock with PIC 16F877A Microcontroller
Date: 23.09.2024	

Aim

To Design and Implement a digital clock system using the PIC16F877A microcontroller with a Real Time Clock (RTC) module.

Software Required

1. Code Composer Studio (CCS) compiler to compile and create hex file.
2. Proteus 8.xx is used for the simulation of circuits.
3. PICkit 2 programmer for burning the hex file to PIC microcontroller.

Hardware Required

S No	Apparatus Name	Range	Quantity
1	Bread Board	-	1
2	Regulated Power Supply	5V, 2A	1
3	PIC 16F877A	40 pin PDIP	1
4	Crystal Oscillator	4 MHz	1
5	PIC Programmer	-	1
6	Liquid Crystal Display	-	1
7	IC DS1307	-	1
8	Quartz Crystal	32.768 KHz	1
9	Resistor	1 KΩ	3
10	Connecting Wires	Single Strand	as required

Procedure

- Connect the SDA and SCL pins of the RTC to the designated I/O pins on the PIC for I2C communication (RC4 for SDA and RC3 for SCL).
- For LCD Connect the control pins (RS, RW, E) to appropriate GPIO pins, and the data pins to PORTD or other available pins.
- Compile the code and load it onto PIC16F877A using a programmer.
- After uploading the program, test the functionality by checking the displayed time.

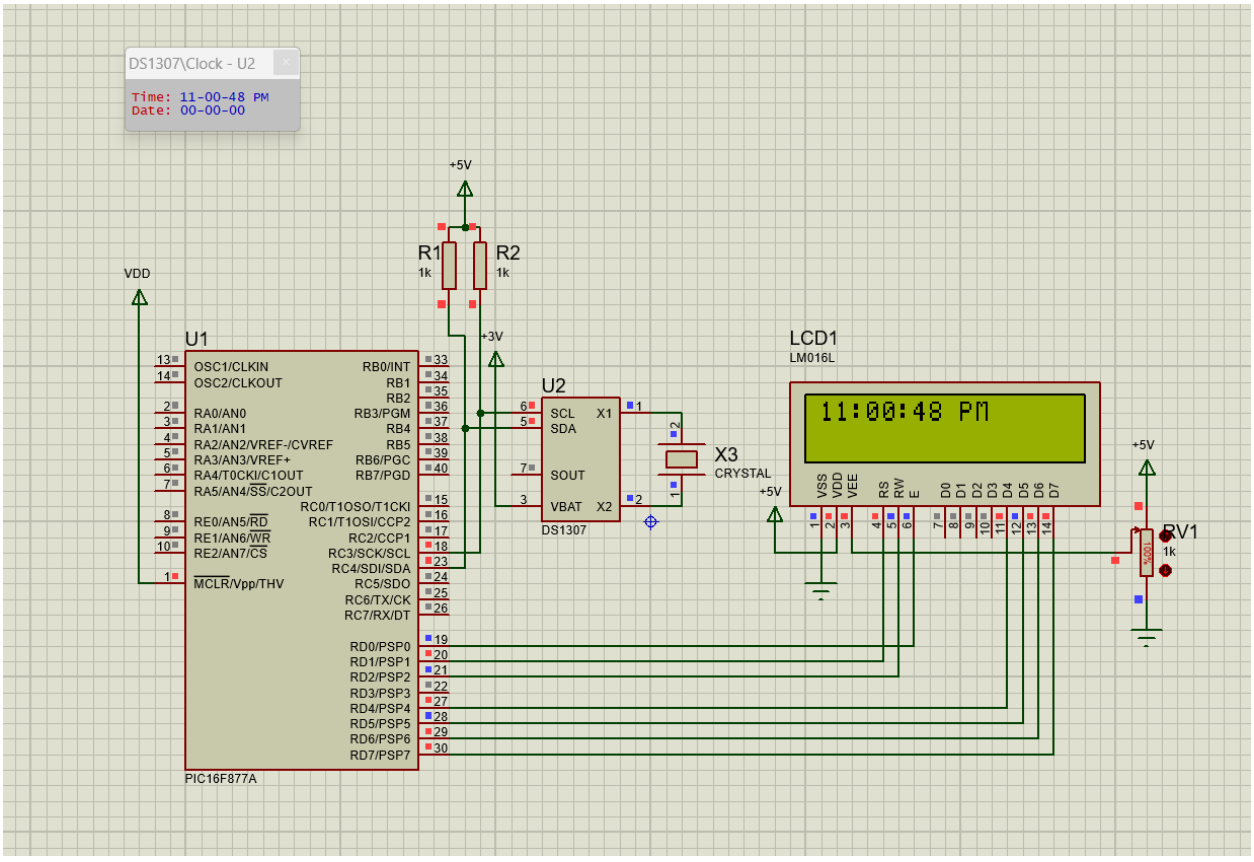
Code

```
#include <16F877A.h>
#device ADC=16
#FUSES NOWDT                    //No Watch Dog Timer
#FUSES PUT                      //Power Up Timer
#FUSES NOBROWNOUT               //No brownout reset
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#use delay(crystal=4MHz)
#use i2c(Master,Fast,sda=PIN_C4,scl=PIN_C3)
#define LCD_ENABLE_PIN PIN_D0
#define LCD_RS_PIN PIN_D1
#define LCD_RW_PIN PIN_D2
#define LCD_DATA4 PIN_D4
#define LCD_DATA5 PIN_D5
#define LCD_DATA6 PIN_D6
#define LCD_DATA7 PIN_D7
#include <lcd.c>
int sec=0x40,min=0x00,hr=0x71,hrs,am_pm;

void write(){
  i2c_start();
  i2c_write(0xD0);
  i2c_write(0x00);
  i2c_write(sec);
  i2c_write(min);
  i2c_write(hr);
  i2c_stop();
}
```

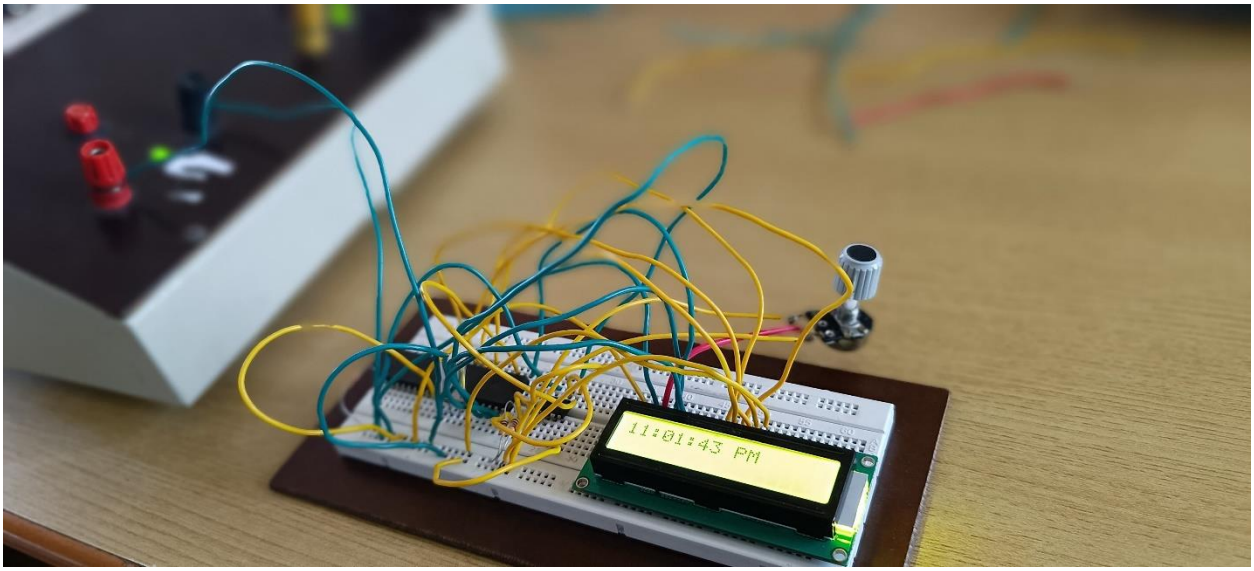
```
byte read(byte add){
  i2c_start();
  i2c_write(0xD0);
  i2c_write(add);
  i2c_stop();
  i2c_start();
  i2c_write(0xD1);
  int data = i2c_read(0);
  i2c_stop();
  return data;
}

void main()
{
  lcd_init();
  write();
  while(TRUE)
  {
    sec=read(0);
    min=read(1);
    hr=read(2);
    hrs=hr&(0x1F);
    lcd_gotoxy(4,1);
    printf(lcd_putc,"\f%X:%X:%X",hrs,min,sec);
    am_pm = hr & (0x20);
    if (am_pm == 0x20) {
      printf(lcd_putc, " PM");
    } else {
```

```
        printf(lcd_putc, " AM");
    }
    delay_ms(250);
  }
}
```

Simulation Output



Hardware Output



QR code

Rubrics		Marks
Conduct of Experiment (20)	Analyse the problem and develop programming constructs (15)	
	Completeness of the experiment (5)	
Observation/ Record (30)	Interpretation of the findings (15)	
	Simulation and Hardware (5)	
	Adherence to record submission deadline (5)	
	Presentation and completion of record (5)	
Viva (10)	Ability to recall the theoretical concepts	
Total (60)		

Result

Thus the Design and Implementation of a digital clock system using the PIC16F877A microcontroller with a Real Time Clock (RTC) module has been done successfully.

Exp No: 5	**Control LEDs via Classic Bluetooth using the WDM board (Bluetooth Communication)**
Date : 30 - 09 - 2024	

Aim

To design and implement a code to control and blink led via Classic Bluetooth using the WDM board and ThingZKit Mini.

Software Required and Hardware Required

- Arduino IDE (Software)
- ThingZMate Kit (Hardware)

Procedure

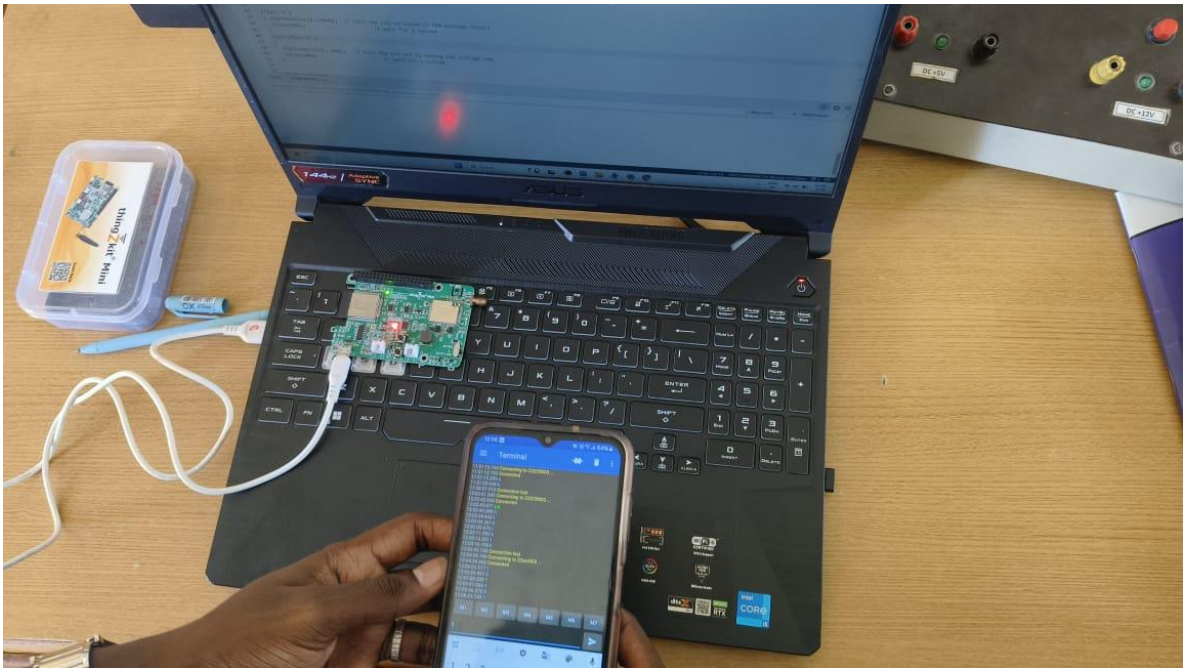
- Open the Arduino IDE
- Click New File and Enter the program as required
- Connect the Kit with the Laptop
- Select ESP 32 WROOM
- Compile and Run the program
- Verify the Output

Program

```
#include "BluetoothSerial.h"
#define LED_BUILTIN 2
//#define USE_PIN // Uncomment this to use PIN during pairing. The pin is specified on the line below
const char *pin = "1234"; // Change this to more secure PIN.
char val;
String device_name = "ESI_LAB";
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif
```

```
#if !defined(CONFIG_BT_SPP_ENABLED)
#error Serial Bluetooth not available or not enabled. It is only available for the ESP32 chip.
#endif
BluetoothSerial SerialBT;

void setup() {
  Serial.begin(115200);
  SerialBT.begin(device_name); //Bluetooth device name
  Serial.printf("The device with name \"%s\" is started.\nNow you can pair it with Bluetooth!\n", device_name.c_str());
  //Serial.printf("The device with name \"%s\" and MAC address %s is started.\nNow you can pair it with Bluetooth!\n", device_name.c_str(), SerialBT.getMacString()); // Use this after the MAC method is implemented
  #ifdef USE_PIN
    SerialBT.setPin(pin);
    Serial.println("Using PIN");
  #endif
}
void loop() {
  if (SerialBT.available()) {
    val=SerialBT.read();
    if(val=='H'){digitalWrite(LED_BUILTIN, HIGH);}
    else if(val=='L'){digitalWrite(LED_BUILTIN, LOW);}
    else{
      digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
      delay(1000);                      // wait for a second
      digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
      delay(1000);
    }
  }
  delay(20);
}
```

Hardware Output



Video References

Rubrics		Marks
Conduct of Experiment (20)	Analyse the problem and develop programming constructs (15)	
	Completeness of the experiment (5)	
Observation/Record (30)	Interpretation of the findings (15)	
	Simulation and Hardware (5)	
	Adherence to record submission deadline (5)	
	Presentation and completion of record (5)	
Viva (10)	Ability to recall the theoretical concepts	
Total(60)		

Result

Thus the designing and implement of a code to control and blink led via classic bluetooth using the WDM board and ThingZKit Mini has been done successfully.

EXP NO: 6	Push sensor data to a cloud platform and create a dashboard via the HTTP protocol using the WDM board (Wifi Communication)
DATE: 21 - 10 - 2024	

Aim

To Push sensor data to a cloud platform and create a dashboard via the HTTP protocol using the WDM board.

Software Required and Hardware Required

- Arduino IDE (Software)
- ThingZMate Kit (Hardware)

Procedure

- Open the Arduino IDE
- Click New File and Enter the program as required
- Connect the Kit with the Laptop
- Select ESP 32 WROOM
- Compile and Run the program
- Verify the Output

Program

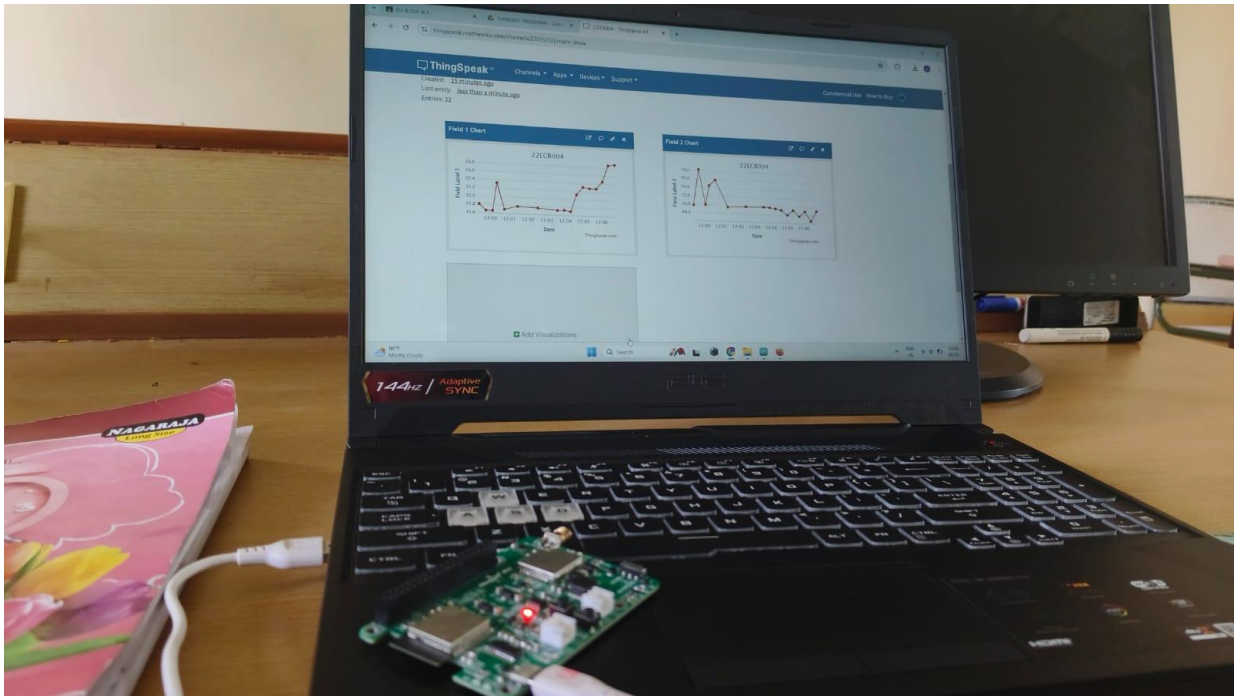
```
#include <Arduino.h>
#include <Wire.h>
#include "Adafruit_SHT31.h"
#include <WiFi.h>
#include <WiFiMulti.h>
#include <HTTPClient.h>
#define USE_SERIAL Serial

bool enableHeater = false;
```

```
uint8_t loopCnt = 0;

Adafruit_SHT31 sht31 = Adafruit_SHT31();
WiFiMulti wifiMulti;

void setup() {
Serial.begin(9600);
while (!Serial)
   delay(10);     // will pause Zero, Leonardo, etc until serial console opens

 Serial.println("SHT31 test");
 if (! sht31.begin(0x44)) {   // Set to 0x45 for alternate i2c addr
  Serial.println("Couldn't find SHT31");
  while (1) delay(1);
 }

 USE_SERIAL.begin(115200);

 USE_SERIAL.println();
 USE_SERIAL.println();
 USE_SERIAL.println();

 for (uint8_t t = 4; t > 0; t--) {
  USE_SERIAL.printf("[SETUP] WAIT %d...\n", t);
  USE_SERIAL.flush();
  delay(1000);
 }

 wifiMulti.addAP("iQOO Z7 Pro 5G", "firefist");
}

void loop() {
```

```
 // wait for WiFi connection
 char buf[300];
 if ((wifiMulti.run() == WL_CONNECTED)) {

  HTTPClient http;

  USE_SERIAL.print("[HTTP] begin...\n");
  // configure traged server and url
  //http.begin("https://www.howsmyssl.com/a/check", ca); //HTTPS

 float t = sht31.readTemperature();
 float h = sht31.readHumidity();

sprintf(buf,"https://api.thingspeak.com/update?api_key=QNIM2DUCOW9D1EA9&field1=%f
&field2=%f",t,h);

  http.begin(buf);  //HTTP

  USE_SERIAL.print("[HTTP] GET...\n");
  // start connection and send HTTP header
  int httpCode = http.GET();

  // httpCode will be negative on error
  if (httpCode > 0) {
   // HTTP header has been send and Server response header has been handled
   USE_SERIAL.printf("[HTTP] GET... code: %d\n", httpCode);

   // file found at server
   if (httpCode == HTTP_CODE_OK) {
    String payload = http.getString();
    USE_SERIAL.println(payload);
   }
```

```
    } else {
      USE_SERIAL.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
    }
    http.end();
  }
  delay(5000);
}
```

Hardware Output



ThingSpeak Cloud Output


```
    entry_id:      12
    field1:        "31.639999"
    field2:        "69.900002"
▼ 12:
    created_at:    "2024-10-21T06:33:35Z"
    entry_id:      13
    field1:        "31.719999"
    field2:        "69.800003"
▼ 13:
    created_at:    "2024-10-21T06:33:57Z"
    entry_id:      14
    field1:        "31.730000"
    field2:        "69.709999"
▼ 14:
    created_at:    "2024-10-21T06:34:18Z"
    entry_id:      15
    field1:        "31.700001"
    field2:        "69.500000"
▼ 15:
    created_at:    "2024-10-21T06:34:39Z"
    entry_id:      16
    field1:        "32.119999"
    field2:        "69.180000"
```

ThingSpeak Channel Output

Video References

Rubrics		Marks
Conduct of Experiment (20)	Analyse the problem and develop programming constructs (15)	
	Completeness of the experiment (5)	
Observation/Record (30)	Interpretation of the findings (15)	
	Simulation and Hardware (5)	
	Adherence to record submission deadline (5)	
	Presentation and completion of record (5)	
Viva (10)	Ability to recall the theoretical concepts	
Total(60)		

Result

Sending the sensor data to cloud using thingzkit mini through wifi was done successfully.

Exp. No.: 7 Date:	Integrate a third-party application server for data storage and monitoring. (LoRaWAN Communication)

Aim:

To design and develop an embedded C program to integrate a third-party application server for data storage and monitoring using LoRaWAN Communication. (Switch status monitoring).

Software Required:

1. Arduino IDE 2.3

Hardware required:

Sl. No	Apparatus name	Range	Quantity
1.	thingZkit Mini	-	1
2.	USB data cable	-	1

Theory:

Building a **thingZmate / The Things Network (TTN) login system** typically refers to creating a way to authenticate devices (LoRaWAN nodes) with thingZmate / The Things Network (TTN) to securely send and receive data over a LoRaWAN network. TTN is an open-source, decentralized IoT network that uses LoRaWAN for communication between low-power devices and gateways. For devices to communicate with TTN, they must authenticate using specific credentials. LoRaWAN in India typically uses the 865 MHz to 867 MHz band for LoRa communications.

Procedure:

Here's a basic step-by-step guide on how to set up a device and authenticate it with **The Things Network (TTN)**:

Assume the LoRa gateway modem is already configured to the TTN network. In this experiment, Class A communication mode and Over-the-Air Activation (OTAA) Authentication is recommended.

Steps to Build a TTN Login (Authentication) System

1. Create a TTN Account

- **Go to the TTN Console**: Visit The Things Network Console. (https://accounts.thethingsindustries.com)
- **Sign up** or log in if you already have an account.

2. Create a New Application in TTN

- Once logged in to the console, navigate to the **Applications** section.
- Click on the **"Add Application"** button.
- Provide a **name** and **description** for your application.
- Select the **Region** that best matches your location (e.g., India).
- After creating the application, you'll be taken to the application dashboard.

3. Create a New Device (Join)

- Inside the application dashboard, go to the **Devices** section.
- Click on **"Add Device"** to register your LoRaWAN device.
- You'll be prompted to enter details like the **Device ID** (a unique identifier for the device), **Device EUI**, and **App Key**.

The device can authenticate using one of the following methods:

- **OTAA (Over-the-Air Activation)**: This mode allows for the device to join the network dynamically. The device will authenticate automatically using the **Device EUI**, **Application EUI**, and **App Key**.

3.1 Using OTAA Authentication (Recommended for most cases)

- **Device EUI**: This is a unique identifier for your device, usually printed on the device itself or available through your device's firmware.
- **Application EUI**: TTN generates this when you create the application. You will find it in the TTN Console under the application's settings.
- **App Key**: This is a secret key used to authenticate the device during OTAA. It's also available in the TTN Console in the application settings.
- Once these values are added to the TTN device, TTN will manage the authentication automatically when your device attempts to connect.

4. Configure Your Device (LoRaWAN Module)

- Given program sketch and AT command should be uploaded before doing the following procedure.
- Open serial monitor console of Arduino IDE and configure the baud rate as 115200 and set line ending as newline.
- The next step is to configure LoRaWAN device with the credentials that obtained from TTN. For thingZkit mini WDM module need to:

- Set the **Device EUI** and **App Key** (for OTAA)

AT Commands:

1. AT Commands needed at Initial to store Keys via ESP32 serial port.

➢ AT Commands to set initially (Mandatory):

• **FFD** // To do factory data reset

• **C01** // To set message type (C01-Conformed/C00-Unconfirmed)

• **R01** // To enable the ADR

• **T1440** // To set the default sampling interval as 1440 minutes (24Hrs) - (Should not give below 5 minutes)

➢ If you set the NJM-N01 you should follow the given steps below:

• **N01** // To set the Activation Type OTAA Mode

• **DXXXXXXXXXXXXXXXX (Device EUI)** // To set Device EUI key

• **JXXXXXXXXXXXXXXXX (App EUI)** // To set APP EUI key

• **AXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX (AppKey)**// To set APP Key

• **ZFF (ATZ)** // To take effective action on below settings (As like saving)

CIRCUIT DIAGRAM:

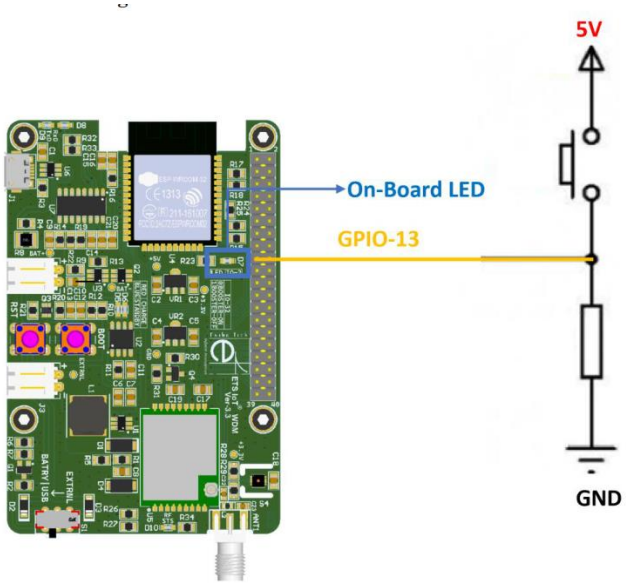
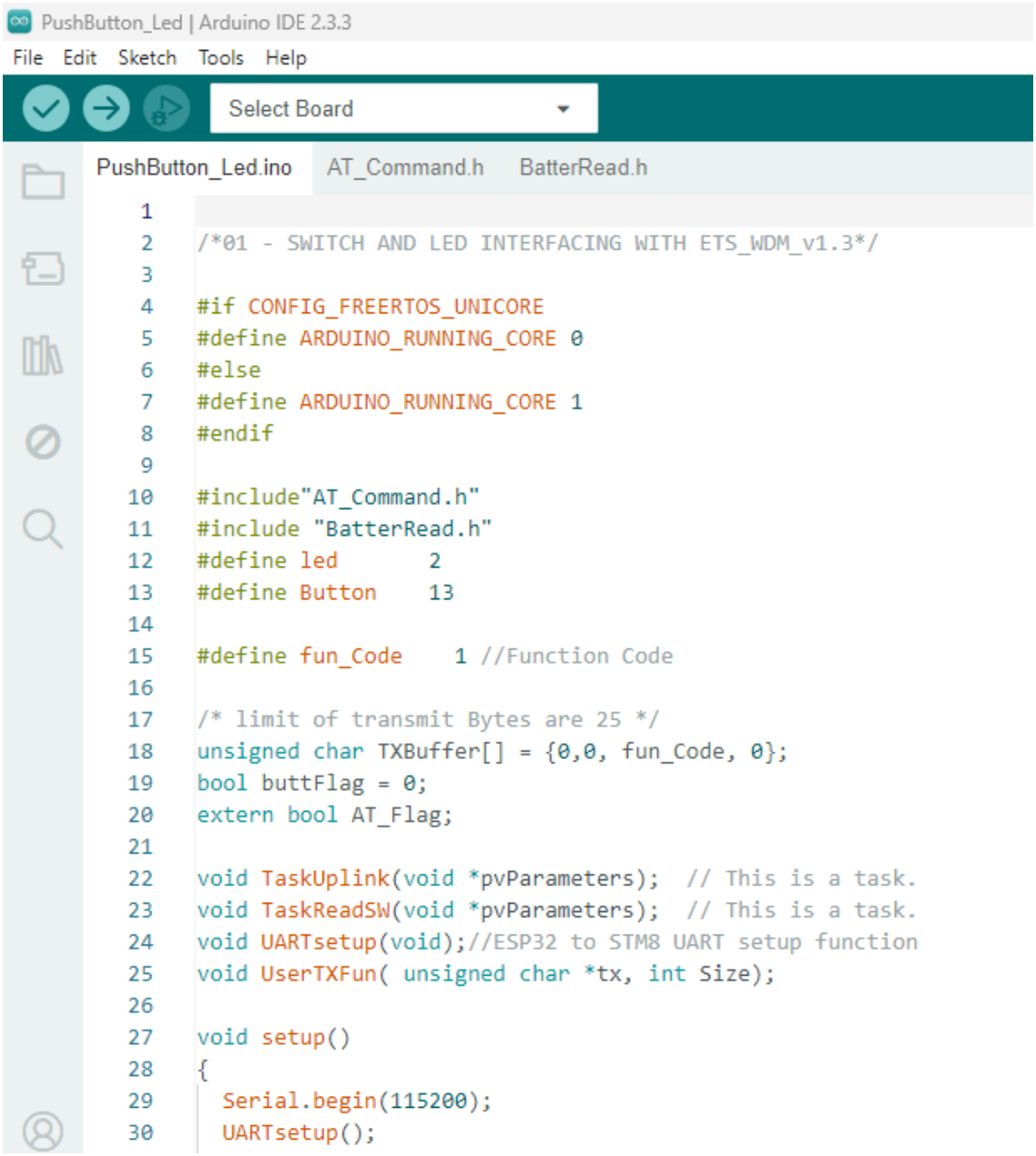


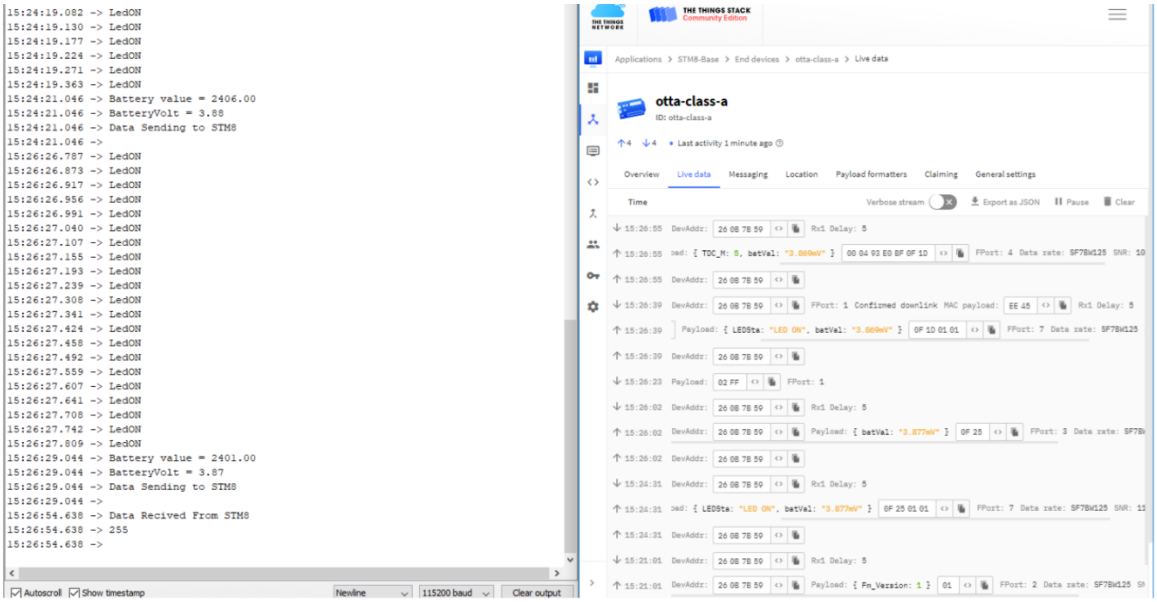
Fig1: SWITH & LED interfacing with ETS_IOT_WDM_v3.3/ ESP32

Program:

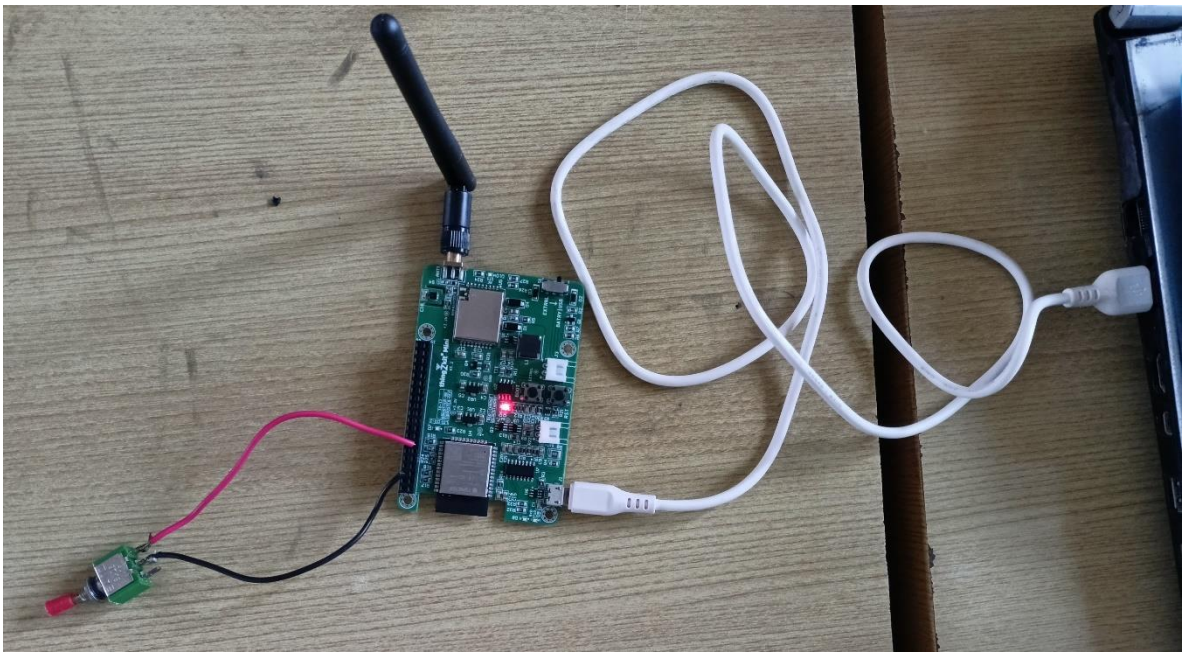

PushButton_Led | Arduino IDE 2.3.3
File Edit Sketch Tools Help
Select Board
PushButton_Led.ino
AT_Command.h
BatterRead.h
1
/*01 - SWITCH AND LED INTERFACING WITH ETS_WDM_v1.3*/
3
#if CONFIG_FREERTOS_UNICORE
#define ARDUINO_RUNNING_CORE 0
#else
#define ARDUINO_RUNNING_CORE 1
#endif
9
#include"AT_Command.h"
#include "BatterRead.h"
#define led 2
#define Button 13
14
#define fun_Code 1 //Function Code
16
/* limit of transmit Bytes are 25 */
unsigned char TXBuffer[] = {0,0, fun_Code, 0};
bool buttFlag = 0;
extern bool AT_Flag;
21
void TaskUplink(void *pvParameters); // This is a task.
void TaskReadSW(void *pvParameters); // This is a task.
void UARTsetup(void);//ESP32 to STM8 UART setup function
void UserTXFun(unsigned char *tx, int Size);
26
void setup()
{
Serial.begin(115200);
UARTsetup();

```
  xTaskCreatePinnedToCore(TaskUplink,  "TaskUplink",  10240,  NULL,  4,  NULL,  ARDUINO_RUNNING_CORE);
  xTaskCreatePinnedToCore(TaskReadSW,  "TaskReadSW",  10240,  NULL,  3,  NULL,  ARDUINO_RUNNING_CORE);
  pinMode(led, OUTPUT);//Set pin 02 as digital output pin
  pinMode(Button, INPUT);//Set pin 13 as digital input pin

}

void loop() {

}

void TaskUplink(void *pvParameters)  // This is a task.
{
  (void) pvParameters;
  for (;;)
  {
    if (buttFlag == 1 && AT_Flag != 1 && joinFlag==1) {
        if (Class_A == 1) {
          BatteryRead();//Read the Battery voltage from ESP32
          TXBuffer[0] = BatRed >> 8;
          TXBuffer[1] = BatRed;
        }
      /* Transmit Function From ESP32 to STM8 */
      UserTXFun(TXBuffer, sizeof(TXBuffer));// User TXBuffer and Buffer size...
      for (int i = 0; i < 1000; i++)
      {
        ;
      }
      memset(TXBuffer, 0, sizeof (TXBuffer));
      TXBuffer[2] = fun_Code;//Here add your experiment number...
      buttFlag = 0;
    }
    vTaskDelay(4000);
  }
}

void TaskReadSW(void *pvParameters)  // This is a task.
{
  (void) pvParameters;
  for (;;)
  {
    bool Read = 0;
    Read =  digitalRead(Button);
    if (Read == 1) {
      digitalWrite(led, HIGH);
      Serial.println("LedON");
      TXBuffer[3] = 0x01;
      buttFlag = 1;
    }
    if (Read == 0) {
      digitalWrite(led, LOW);
    }
    vTaskDelay(50);
  }
}
```

Simulation output:



Hardware Output:



Rubrics		Marks
Conduct of Experiment (20)	Analyse the problem and develop programming constructs (15)	
	Completeness of the experiment (5)	
Observation/ Record (30)	Interpretation of the findings (15)	
	Simulation and Hardware (5)	
	Adherence to record submission deadline (5)	
	Presentation and completion of record (5)	
Viva (10)	Ability to recall the theoretical concepts	
Total(60)		

Result:

Thus, the design and development of an embedded C program to integrate a third-party application server for data storage and monitoring using LoRaWAN Communication was done successfully.

Exp. No.: 8 Date:	Integrate a third-party application server for controlling the motor using the WDM board. (LoRaWAN Communication)

Aim:

To design and develop an embedded C program to integrate a third-party application server for controlling the motor using the WDM board using LoRaWAN Communication.

Software Required:

1. Arduino IDE 2.3

Hardware required:

Sl. No	Apparatus name	Range	Quantity
1.	thingZkit Mini	-	1
2.	USB data cable	-	1
3.	DC Stepper Motor	-	1
4.	L298N (or) UL2003 Motor Driver	-	1
5.	DC +12V Power Supply	-	1
6.	Connecting wires	-	As required

Theory:

Building a **thingZmate / The Things Network (TTN) login system** typically refers to creating a way to authenticate devices (LoRaWAN nodes) with thingZmate / The Things Network (TTN) to securely send and receive data over a LoRaWAN network. TTN is an open-source, decentralized IoT network that uses LoRaWAN for communication between low-power devices and gateways. For devices to communicate with TTN, they must authenticate using specific credentials. LoRaWAN in India typically uses the 865 MHz to 867 MHz band for LoRa communications.

Procedure:

Here's a basic step-by-step guide on how to set up a device and authenticate it with **The Things Network (TTN)**:

Assume the LoRa gateway modem is already configured to the TTN network. In this experiment, Class A communication mode and Over-the-Air Activation (OTAA) Authentication is recommended.

Steps to Build a TTN Login (Authentication) System

1. Create a TTN Account

- **Go to the TTN Console**: Visit The Things Network Console. (https://accounts.thethingsindustries.com)
- **Sign up** or log in if you already have an account.

2. Create a New Application in TTN

- Once logged in to the console, navigate to the **Applications** section.
- Click on the **"Add Application"** button.
- Provide a **name** and **description** for your application.
- Select the **Region** that best matches your location (e.g., India).
- After creating the application, you'll be taken to the application dashboard.

3. Create a New Device (Join)

- Inside the application dashboard, go to the **Devices** section.
- Click on **"Add Device"** to register your LoRaWAN device.
- You'll be prompted to enter details like the **Device ID** (a unique identifier for the device), **Device EUI**, and **App Key**.

The device can authenticate using one of the following methods:

- **OTAA (Over-the-Air Activation)**: This mode allows for the device to join the network dynamically. The device will authenticate automatically using the **Device EUI**, **Application EUI**, and **App Key**.

3.1 Using OTAA Authentication (Recommended for most cases)

- **Device EUI**: This is a unique identifier for your device, usually printed on the device itself or available through your device's firmware.
- **Application EUI**: TTN generates this when you create the application. You will find it in the TTN Console under the application's settings.
- **App Key**: This is a secret key used to authenticate the device during OTAA. It's also available in the TTN Console in the application settings.
- Once these values are added to the TTN device, TTN will manage the authentication automatically when your device attempts to connect.

4. Configure Your Device (LoRaWAN Module)

- Given program sketch and AT command should be uploaded before doing the following procedure.
- Open serial monitor console of Arduino IDE and configure the baud rate as 115200 and set line ending as newline.
- The next step is to configure LoRaWAN device with the credentials that obtained from TTN. For thingZkit mini WDM module need to:
    - Set the **Device EUI** and **App Key** (for OTAA)

AT Commands:

1. AT Commands needed at Initial to store Keys via ESP32 serial port.

➢ AT Commands to set initially (Mandatory):

• **FFD** // To do factory data reset

• **C01** // To set message type (C01-Conformed/C00-Unconfirmed)

• **R01** // To enable the ADR

• **T1440** // To set the default sampling interval as 1440 minutes (24Hrs) - (Should not give below 5 minutes)

➢ If you set the NJM-N01 you should follow the given steps below:

• **N01** // To set the Activation Type OTAA Mode

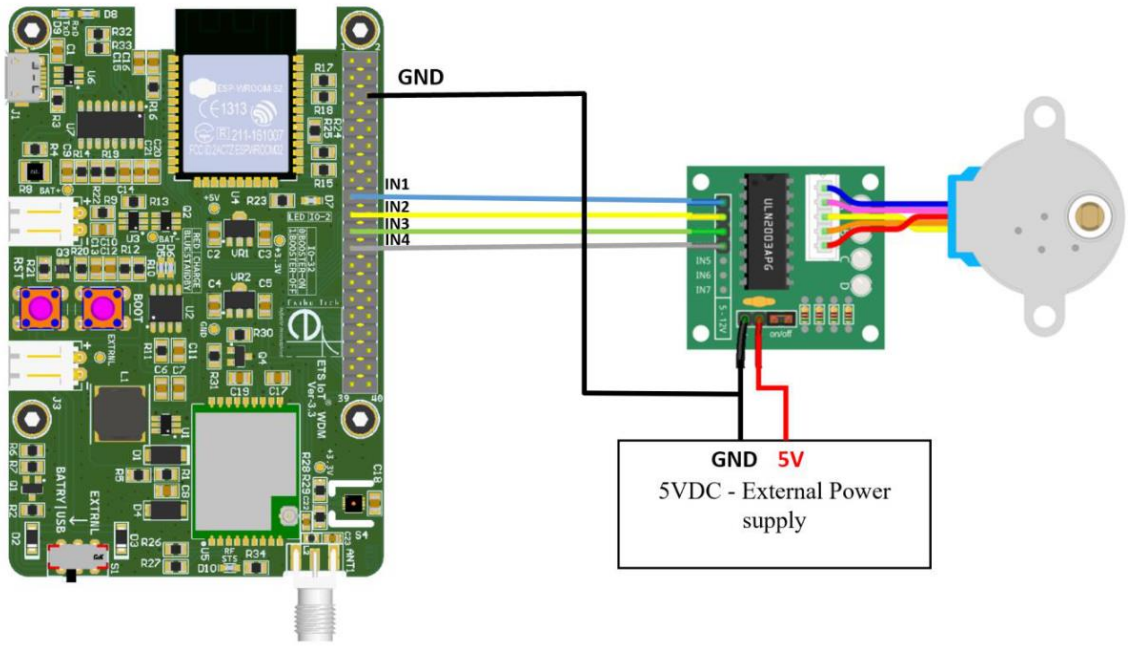
• **DXXXXXXXXXXXXXXXX (Device EUI)** // To set Device EUI key

• **JXXXXXXXXXXXXXXXX (App EUI)** // To set APP EUI key

• **AXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX (AppKey)**// To set APP Key

• **ZFF (ATZ)** // To take effective action on below settings (As like saving)

CIRCUIT DIAGRAM:



Program:

Stepper_Motor.ino | AT_Command.h | BatterRead.h

```
/*Note: use the 5v externalpower supply*/

#if CONFIG_FREERTOS_UNICORE
#define ARDUINO_RUNNING_CORE 0
#else
#define ARDUINO_RUNNING_CORE 1
#endif
#define fun_Code    14 //Function Code

#include "AT_Command.h"
#include "BatterRead.h"
#include <Stepper.h> // Include the header file
#define RX        16
#define TX        17
#define IN1       13
#define IN2       12
#define IN3       14
#define IN4       15

// change this to the number of steps on your motor
#define STEPS 2000

// create an instance of the stepper class using the steps and pins
Stepper stepper (STEPS, IN1, IN2, IN3, IN4); // Create instances in w

unsigned char TXBuffer[] = {0, 0, fun_Code};
uint8_t sendCount = 0;
/*set the flag after received stm8 data*/
bool DowFlag = 0, EnableFlag = 0;
uint8_t RREnCont = 0; // For delay purpose variable
uint16_t value = 0;// For store the Downlink value in this variable

/* RTOS Function */
void TaskDownlink(void *pvParameters);
void TaskStepper(void *pvParameters);

/* User Fucntion */
void StepperRun(uint16_t rotaVal);

void setup()
{
  Serial.begin(115200);
  UARTsetup();
  xTaskCreatePinnedToCore(TaskStepper,  "TaskStepper",  10240,  NULL,  4,  NULL,  ARDUINO_RUNNING_CORE);
  xTaskCreatePinnedToCore(TaskDownlink,  "TaskDownlink",  10240,  NULL,  3,  NULL,  ARDUINO_RUNNING_CORE);

  /*No uplink & Frist time pass the 0 value */
  unsigned char txValue = 0;
  UserTXFun(&txValue, 1);

}

void loop() {

}

void TaskStepper(void *pvParameters)  // This is a task.
{
  (void) pvParameters;
  for (;;)
  {
    if (Class_A == 1) {
      sendCount++;
    }
    if (sendCount == 60) {
      BatteryRead();//Read the Battery voltage from ESP32
      TXBuffer[0] = BatRed >> 8;
```

Stepper_Motor.ino | AT_Command.h | BatterRead.h

```
void loop() {

}

void TaskStepper(void *pvParameters)  // This is a task.
{
  (void) pvParameters;
  for (;;)
  {
    if (Class_A == 1) {
      sendCount++;
    }
    if (sendCount == 60) {
      BatteryRead();//Read the Battery voltage from ESP32
      TXBuffer[0] = BatRed >> 8;
      TXBuffer[1] = BatRed;
      UserTXFun(TXBuffer, sizeof(TXBuffer));// User TXBuffer and Buffer size...
      memset(TXBuffer, 0, sizeof (TXBuffer));
      TXBuffer[2] = fun_Code;
      sendCount = 0;
    }
    StepperRun( value );
    vTaskDelay(1000);
  }
}

void TaskDownlink(void *pvParameters)
{

  (void) pvParameters;
  for (;;)
  {
    /*When the user receives the downlink, the flag will be set.*/
    if (usDwStFlag == 1 && joinFlag == 1) {
      RREnCont++;
    }
```

```
void StepperRun(uint16_t rotaVal) {
  /* One complete rotation given value is 520. you will be achieved the proper angle when you change the given value from a network server. */
  if ( EnableFlag == 1) {
    Serial.println(" Enter Stepper is run ");
    for (int i = 0; i < value; i++)
    {
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, LOW);
      delay(1);
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, HIGH);
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, LOW);
      delay(1);
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, HIGH);
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, LOW);
      delay(1);
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, HIGH);
      digitalWrite(IN3, HIGH);
      digitalWrite(IN4, LOW);
      delay(1);
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, LOW);
      digitalWrite(IN3, HIGH);
      digitalWrite(IN4, LOW);
      delay(1);
```

Stepper_Motor.ino | AT_Command.h | BatterRead.

```
153         digitalWrite(IN2, LOW);
154         digitalWrite(IN3, LOW);
155         digitalWrite(IN4, HIGH);
156         delay(1);
157       }
158       digitalWrite(IN1, HIGH);
159       digitalWrite(IN2, LOW);
160       digitalWrite(IN3, LOW);
161       digitalWrite(IN4, LOW);
162       delay(100);
163       value = 0;
164       EnableFlag = 0;
165
166     } else if (DowFlag == 1 ) {
167       digitalWrite(IN1, LOW);
168       digitalWrite(IN2, LOW);
169       digitalWrite(IN3, LOW);
170       digitalWrite(IN4, LOW);
171       delay(10);
172       DowFlag = 0;
173       EnableFlag = 1;
174     } else {}
175   }
176
```

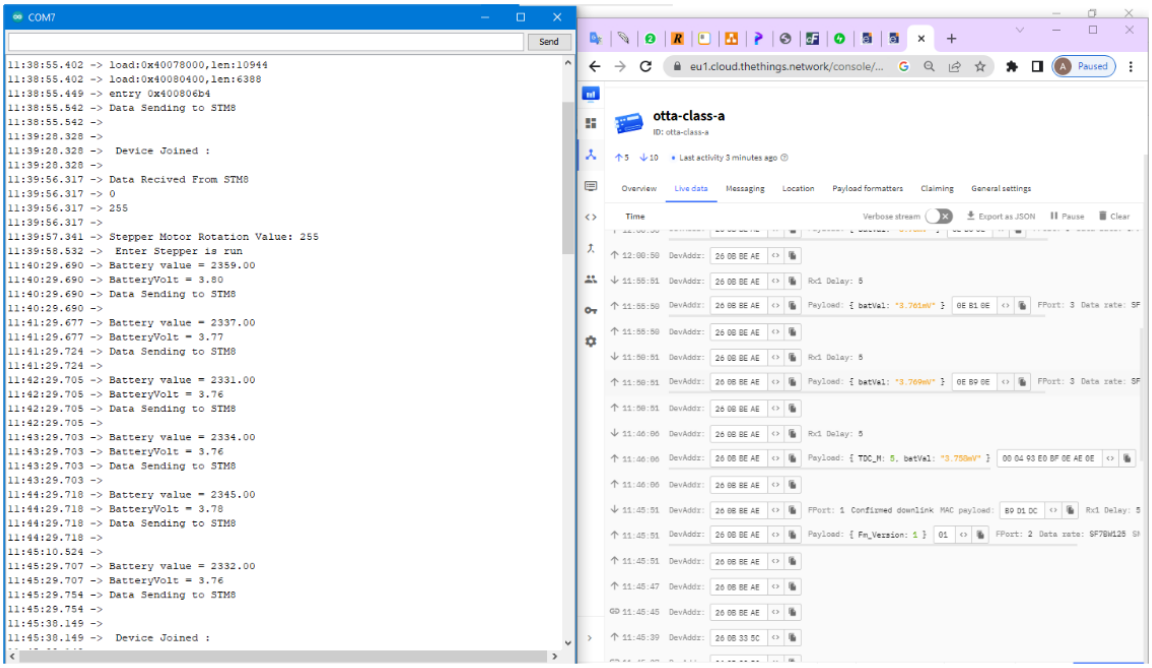
Output | Serial Monitor ×

Message (Enter to send message to 'ESP32 Dev Mod

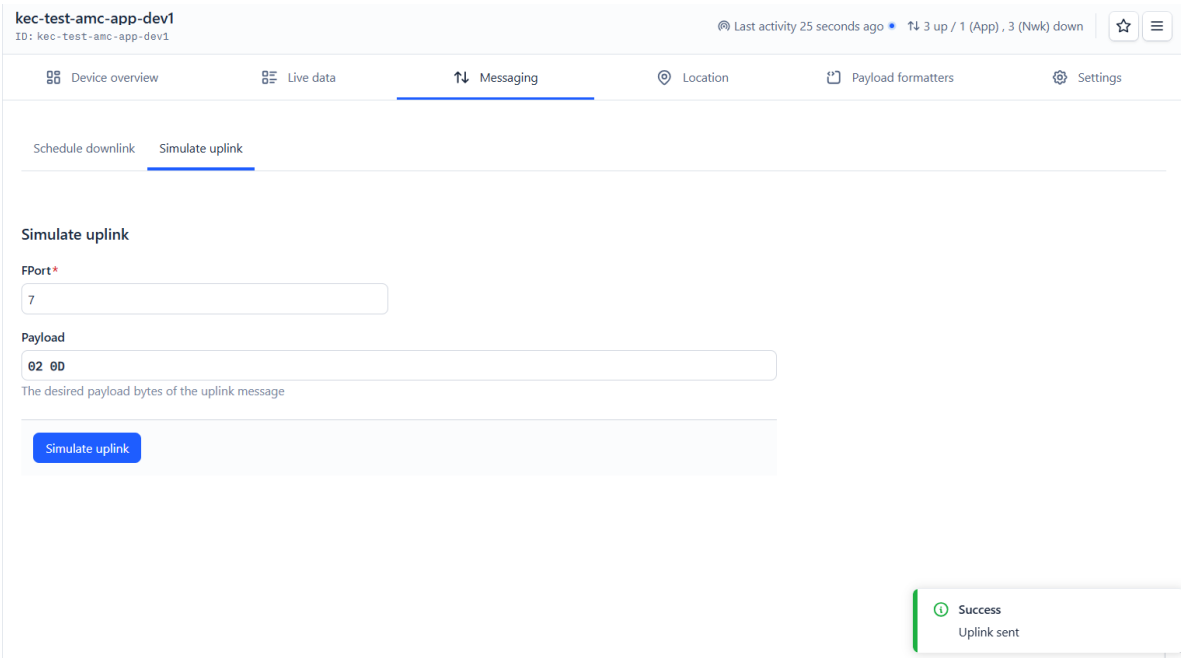
```
load:0x40080400,len:4
load:0x40080404,len:3504
entry 0x400805cc
Data Sending to STM8

Data Recived From STM8
11
```

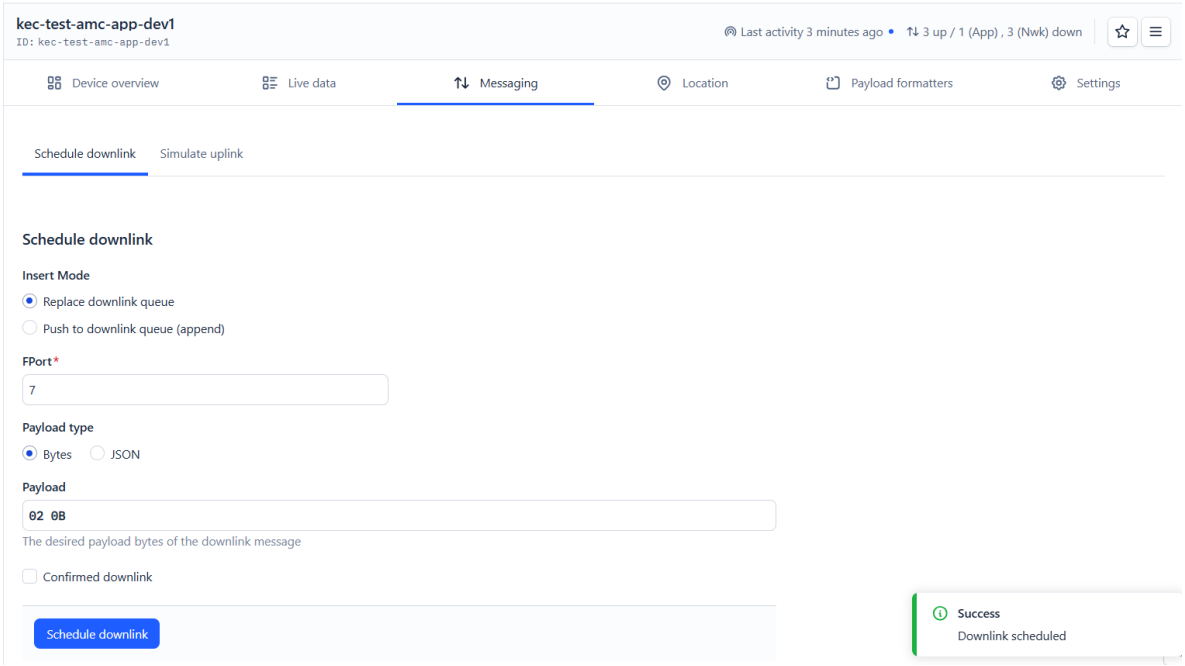
Simulation output:

1. Status Monitoring

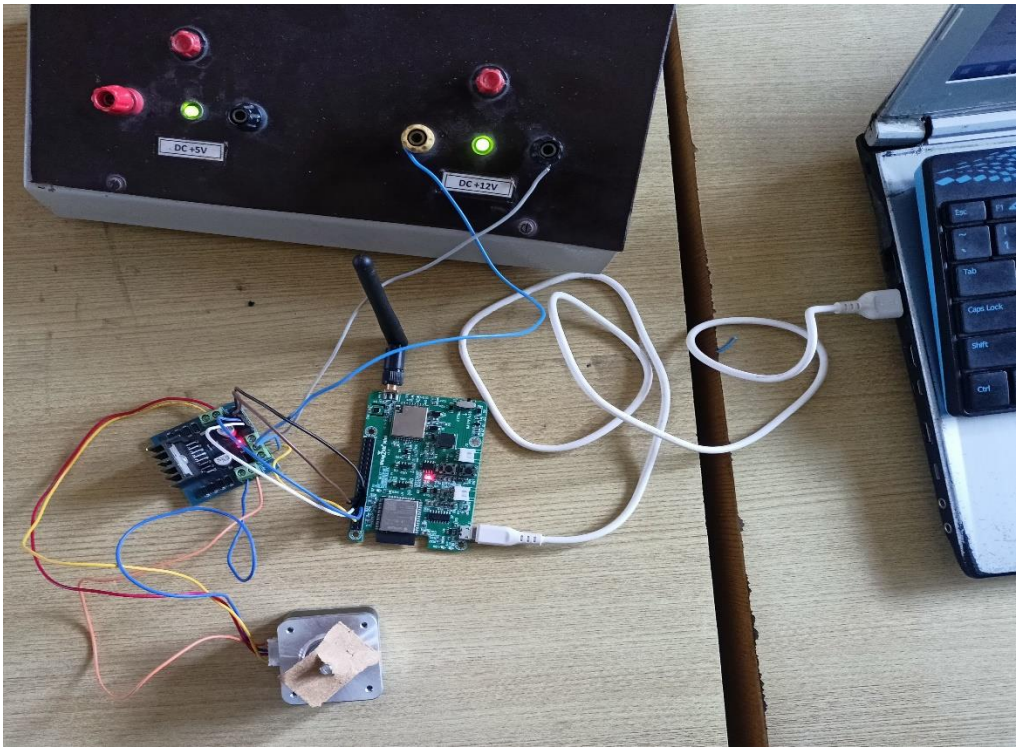


2. Uplink and Downlink to control DC stepper motor from cloud





Hardware Output:



Rubrics		Marks
Conduct of Experiment (20)	Analyse the problem and develop programming constructs (15)	
	Completeness of the experiment (5)	
Observation/ Record (30)	Interpretation of the findings (15)	
	Simulation and Hardware (5)	
	Adherence to record submission deadline (5)	
	Presentation and completion of record (5)	
Viva (10)	Ability to recall the theoretical concepts	
Total(60)		

Result:

Thus, the design and development of an embedded C program to integrate a third-party application server for controlling the motor using the WDM board using LoRaWAN Communication was done successfully.