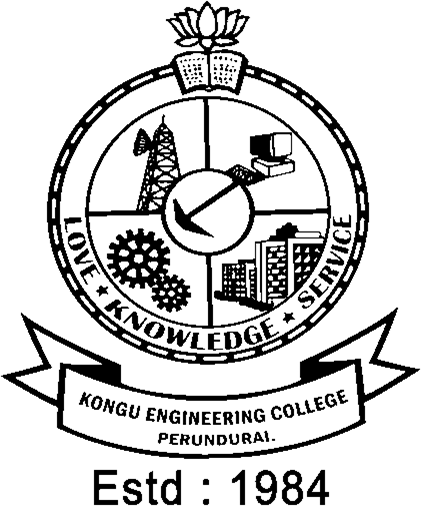
KONGU ENGINEERING COLLEGE

**(Autonomous)**

**Perundurai, Erode– 638060.**

****

**LABORATORY RECORD**

**Name**

**: KISHOREKUMAR S**

**Register Number : 23ECR117 \_**

**Course code**

**Course name**

**: 22ECC51**

**: EMBEDDED SYSTEMS AND IOT**

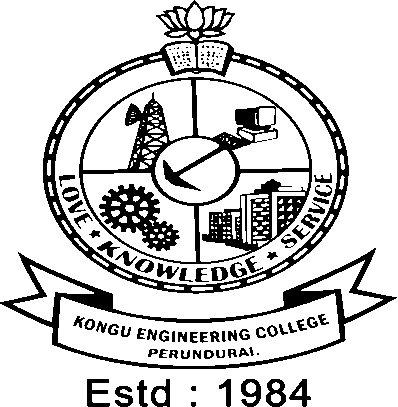
**Semester & Branch : ­­­­­­­V / ECE – B**

****

**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* ***2025–2026.***

**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.
  1. **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)

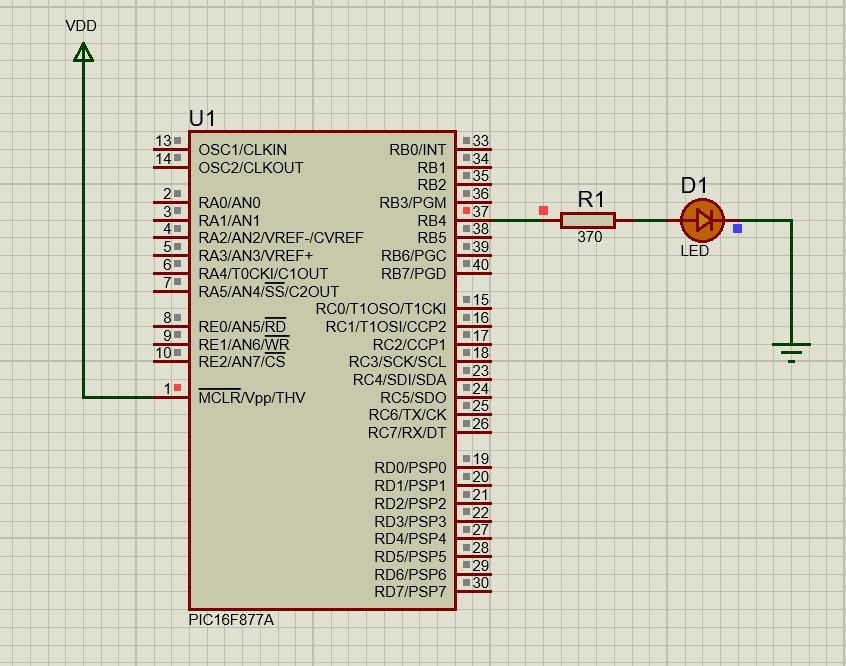
{

//TODO: User Code output\_high(PIN\_B4); delay\_ms(1000); output\_low(PIN\_B4); delay\_ms(1000);

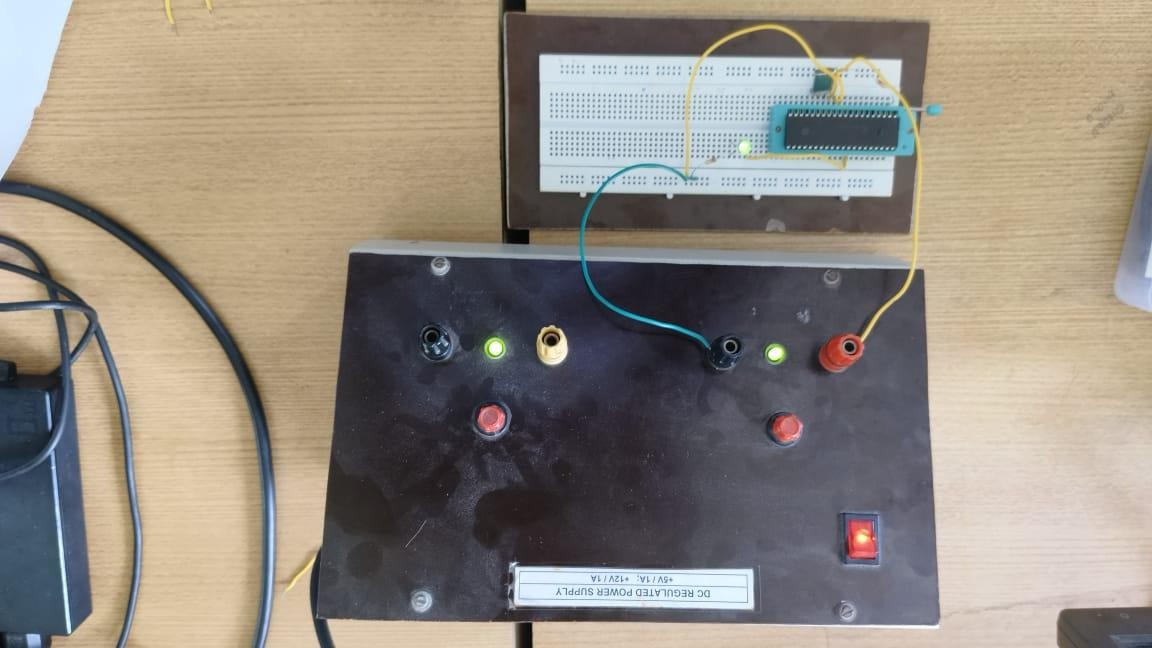
}

}

**Simulation output:**

****

**Hardware Output:**

****

* 1. **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)

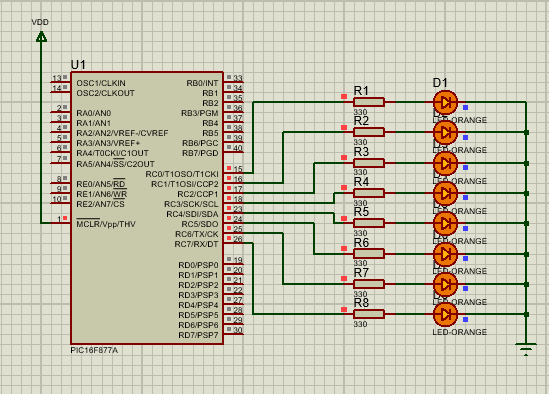
{

output\_C(0x00); delay\_ms(1000); output\_C(0xFF); delay\_ms(1000);

}

}

**Simulation output:**

****

* 1. **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)

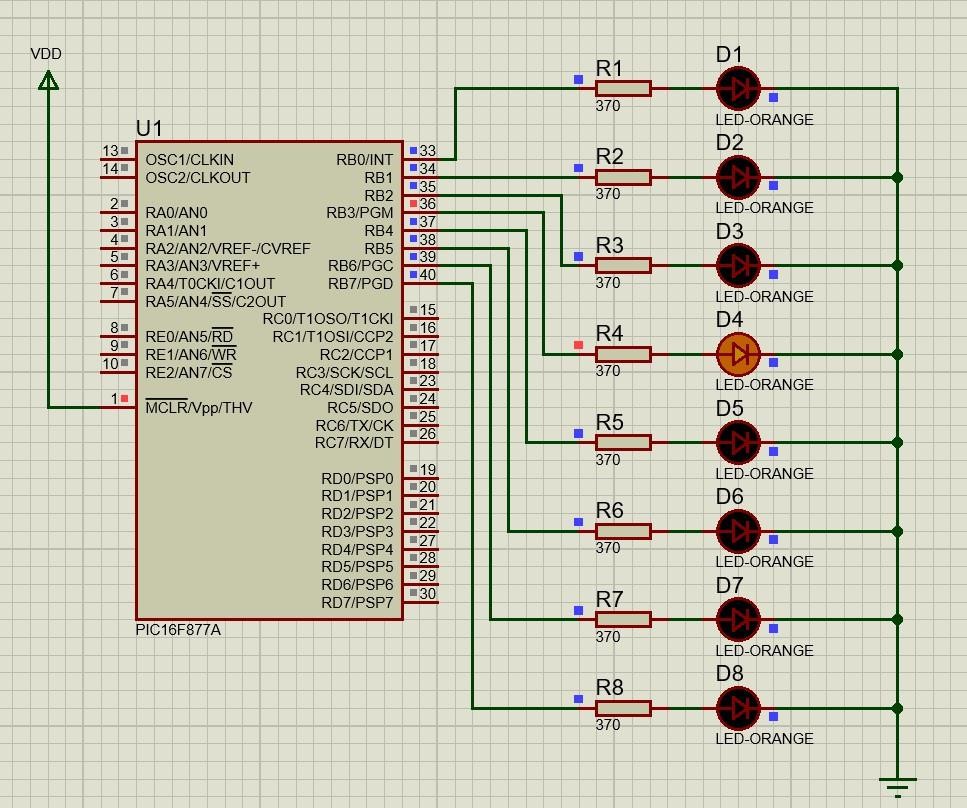
{

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:**

****

* 1. **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);

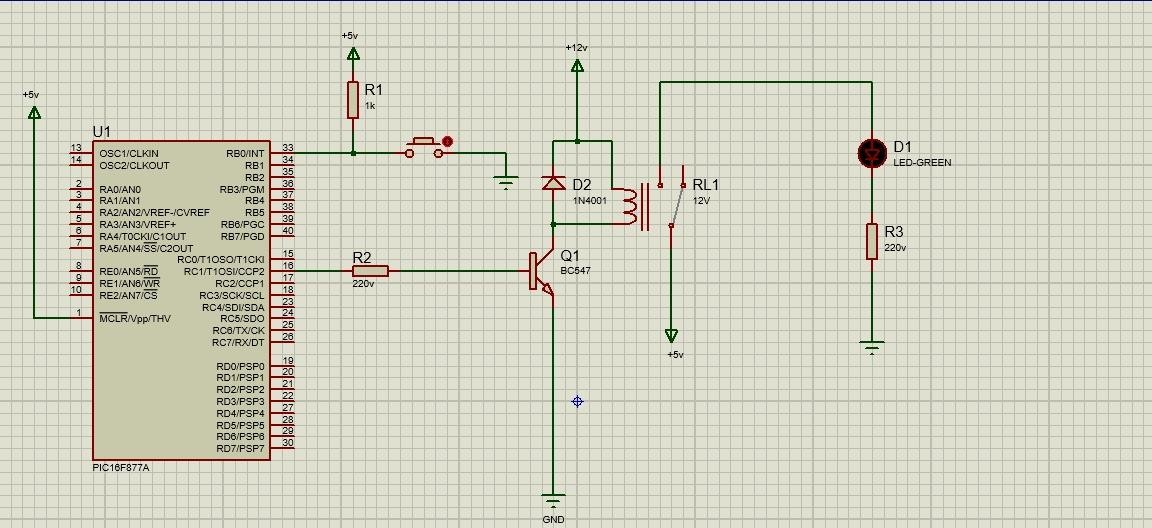
else

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)

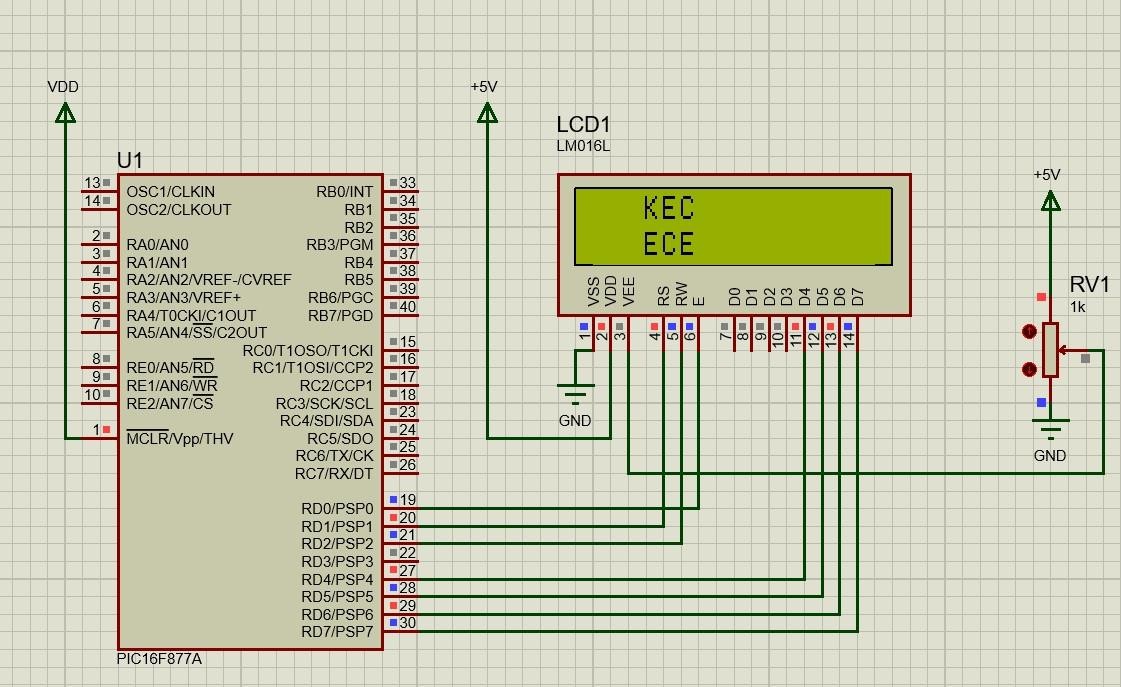
{

lcd\_gotoxy(4,1); lcd\_putc("KEC"); lcd\_gotoxy(4,2); lcd\_putc("ECE"); delay\_ms(800);

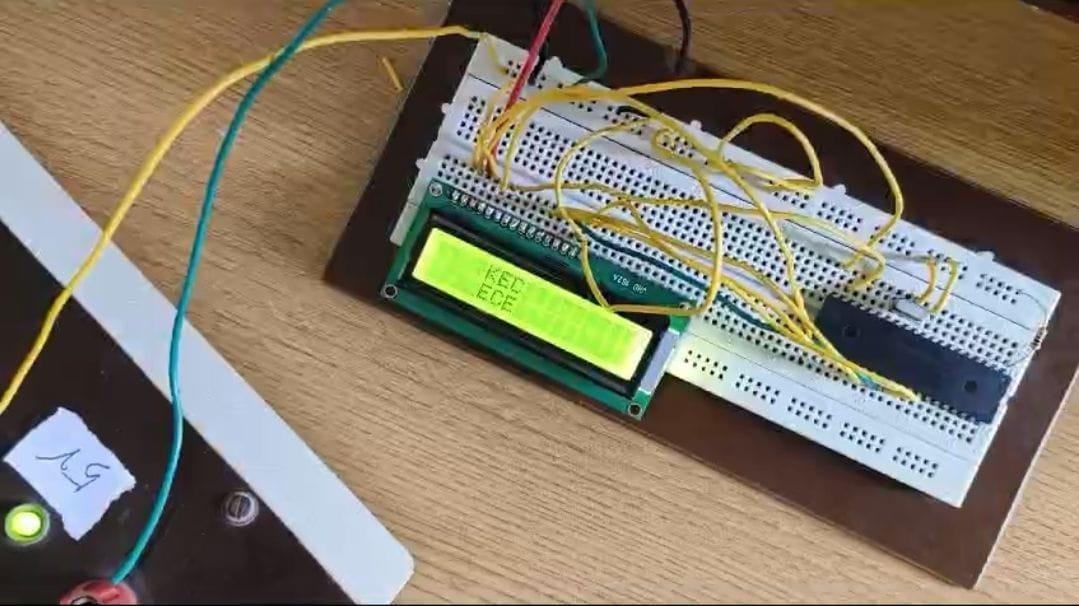
}

}

**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();

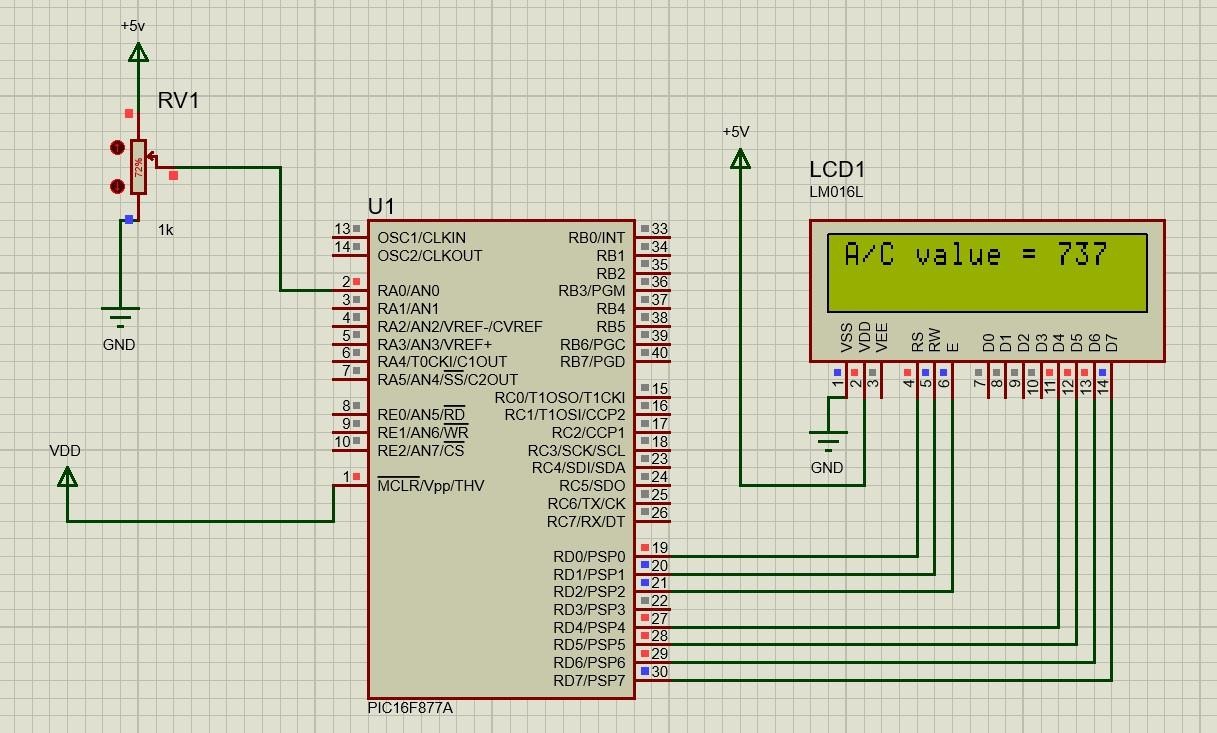
}

value = read\_adc(ADC\_READ\_ONLY); printf(lcd\_putc,"\fA/C value = %ld", value); delay\_ms(500);

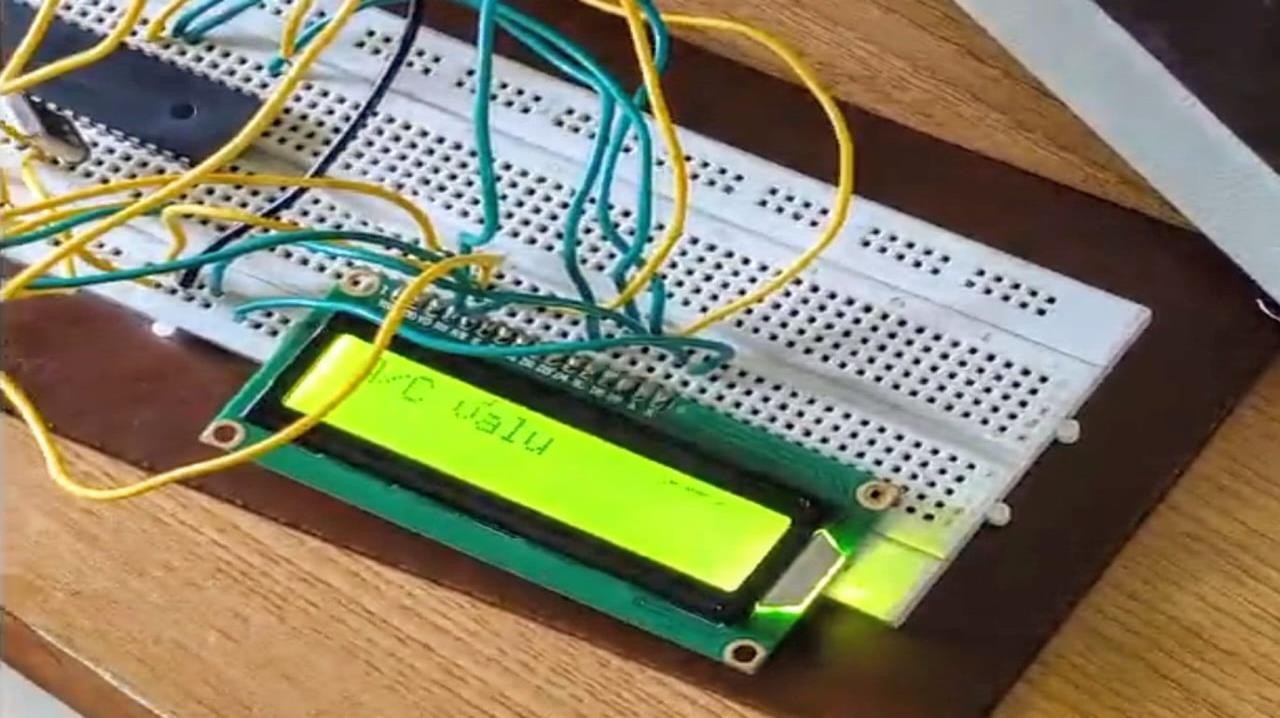
}

}

**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");

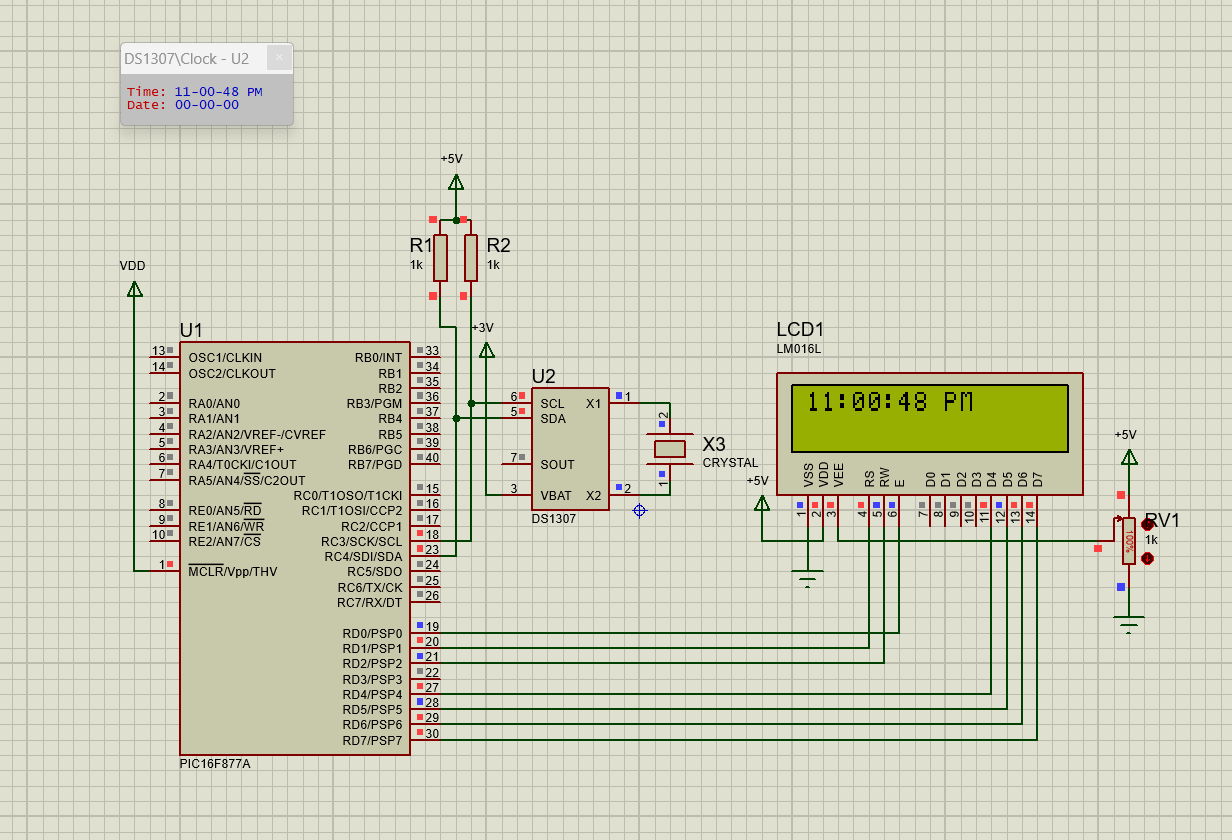
}

delay\_ms(250);

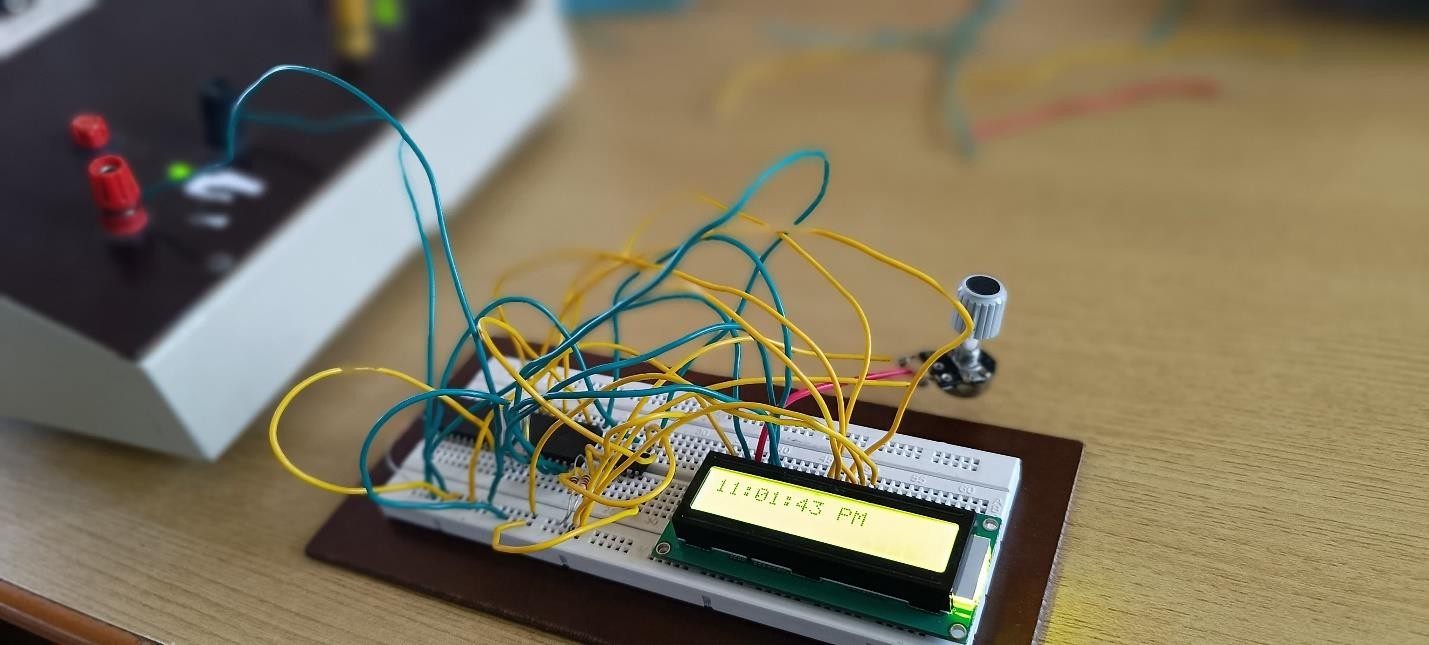
}

}

**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)
* Thing ZMate 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);

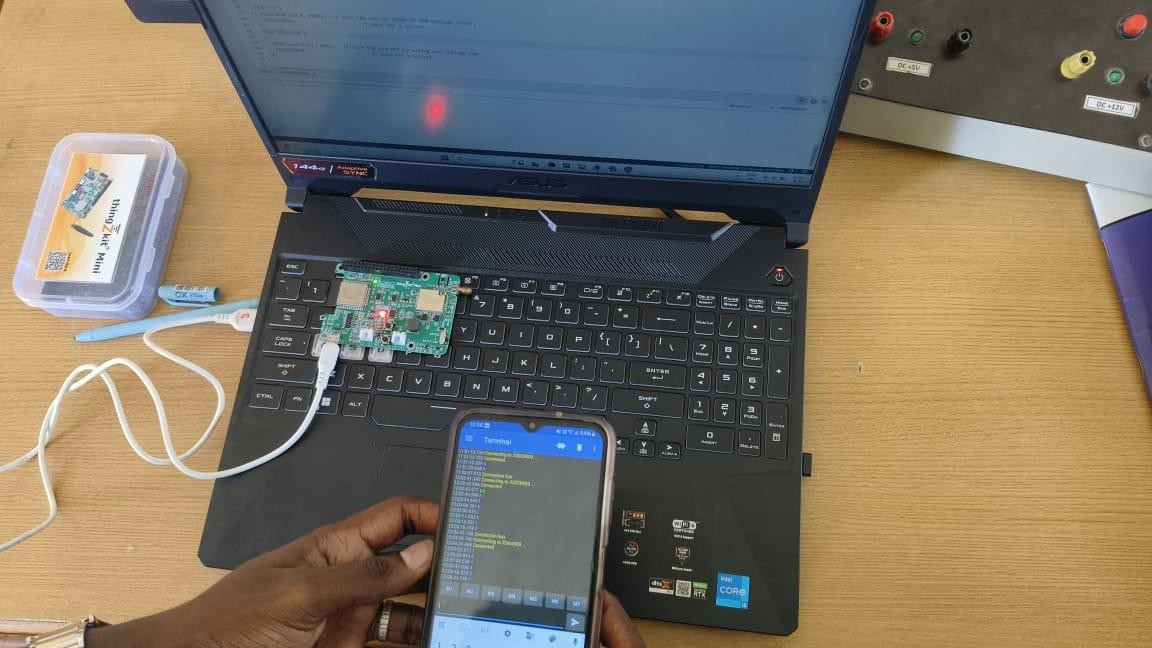
}

}

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](http://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());

}

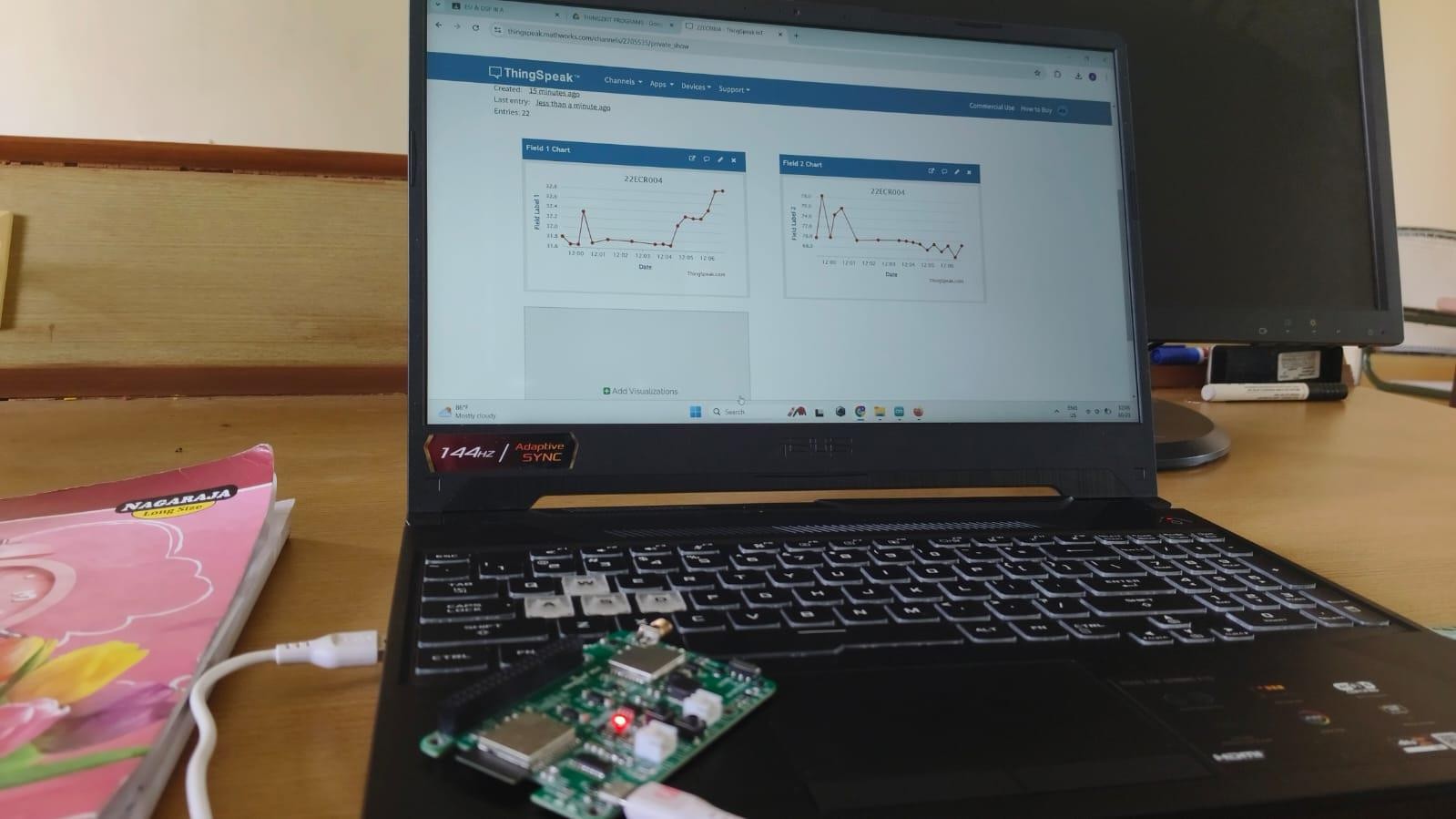
http.end();

}

delay(5000);

}

**Hardware Output**

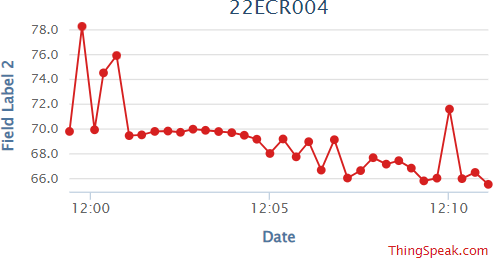
****

**ThingSpeak Cloud Output**

****

**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.

**22ECR004**

|  |  |
| --- | --- |
| **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](https://accounts.thethingsindustries.com/login?from=%2Fdashboard))
* **Sign up** or log in if you already have an account.

1. **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.

1. **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**.
  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.

1. **Configure Your Device (LoRaWAN Module)**

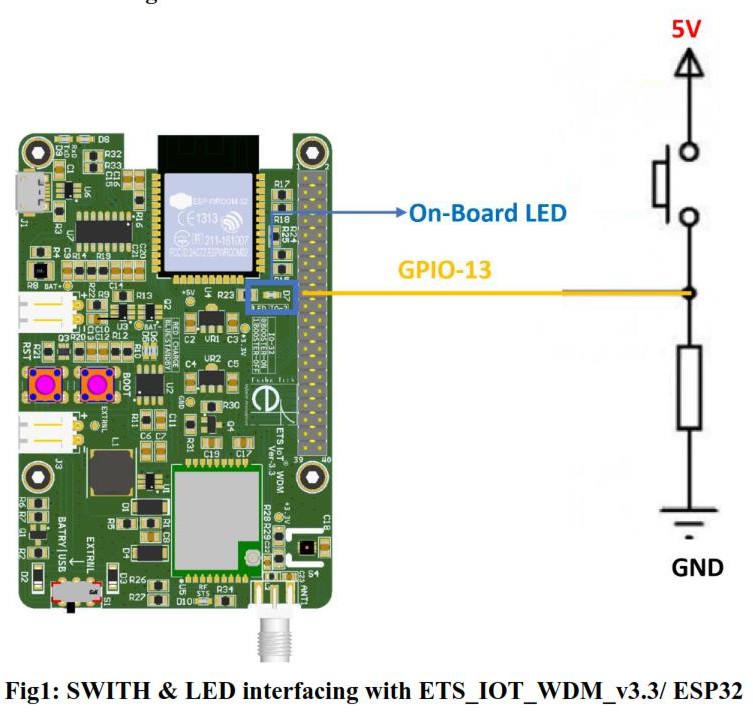
* Given program sketch and AT command should be uploaded before doing the following procedure.
* Open serial monitor console of Arduino IDE and configure the baud rate as 115200 and set line ending as newline.
* The next step is to configure LoRaWAN device with the credentials that obtained from TTN.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

**AT Commands:**

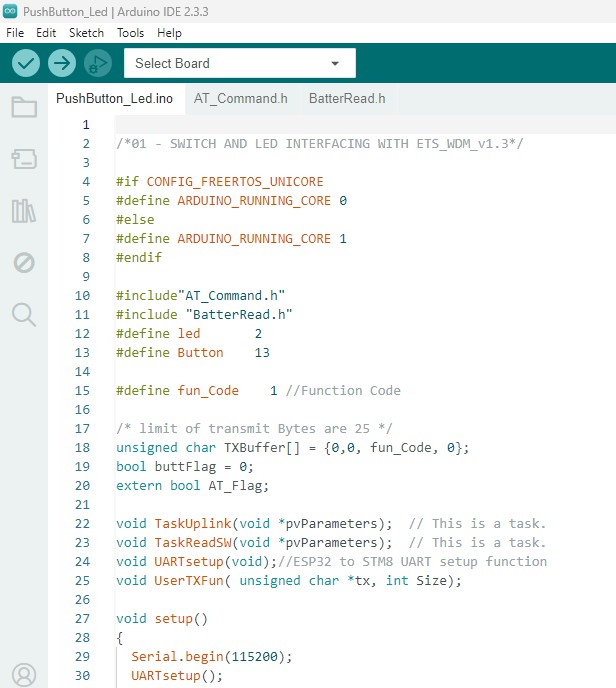
1. AT Commands needed at Initial to store Keys via ESP32 serial port.
   * AT Commands to set initially (Mandatory):

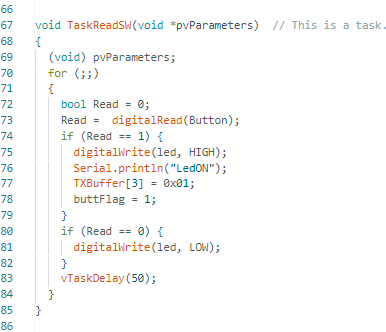
* **FFD** // To do factory data reset
* **C01** // To set message type (C01-Conformed/C00-Unconfirmed)
* **R01** // To enable the ADR
* **T1440** // To set the default sampling interval as 1440 minutes (24Hrs) - (Should not give below 5 minutes)
  + If you set the NJM-N01 you should follow the given 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:**

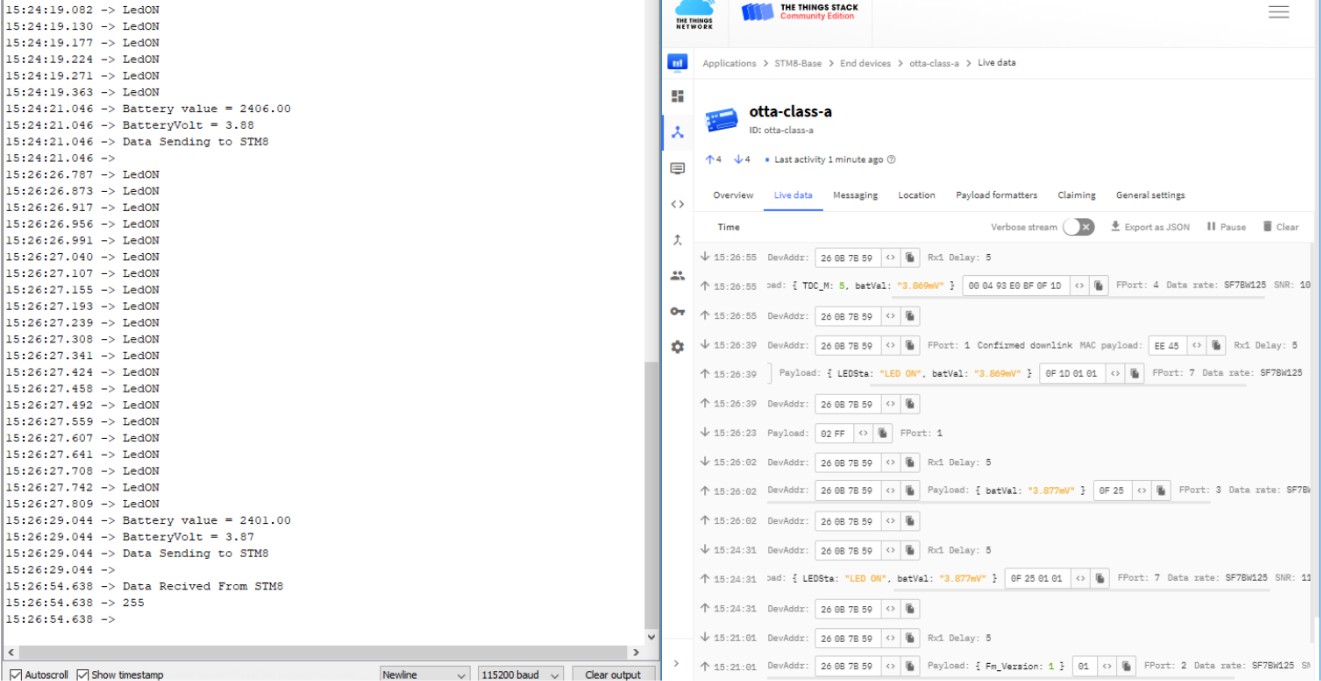
****

**Program:**

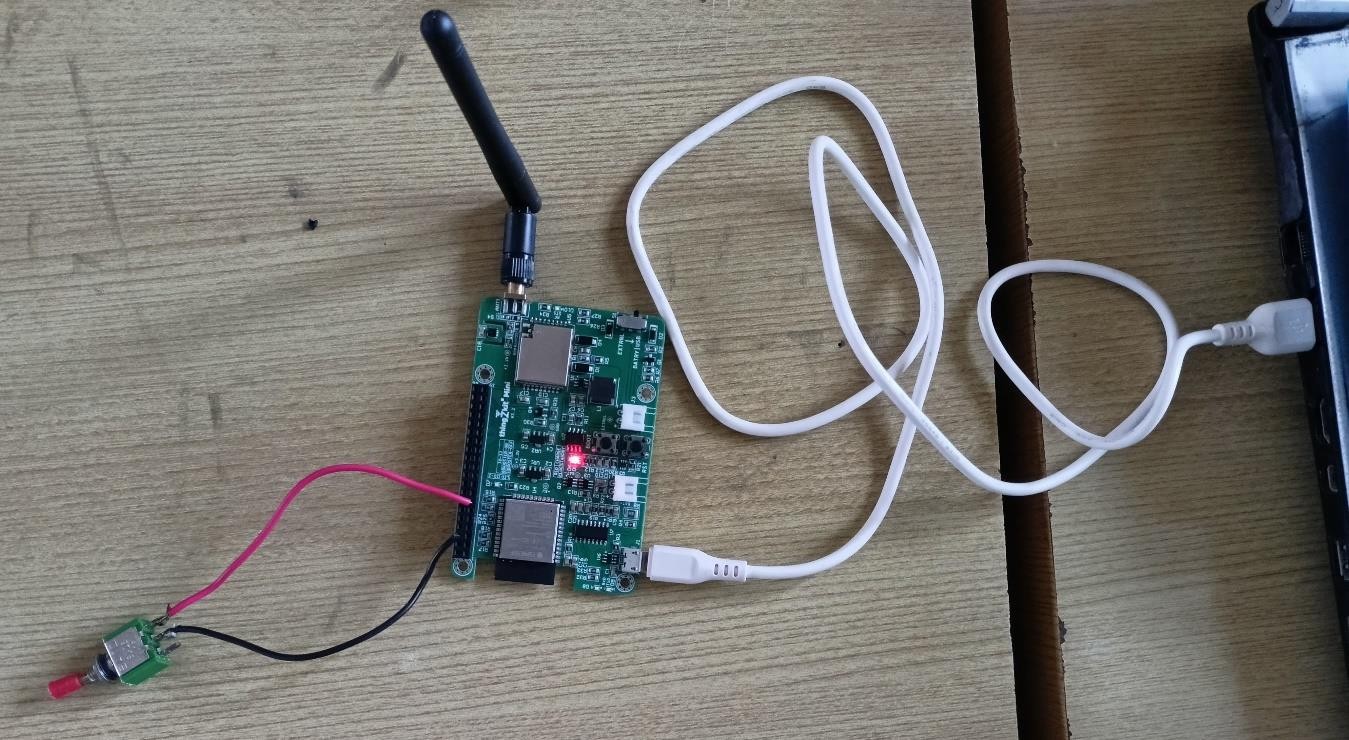
****



**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](https://accounts.thethingsindustries.com/login?from=%2Fdashboard))
* **Sign up** or log in if you already have an account.

1. **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.

1. **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**.
  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.

1. **Configure Your Device (LoRaWAN Module)**

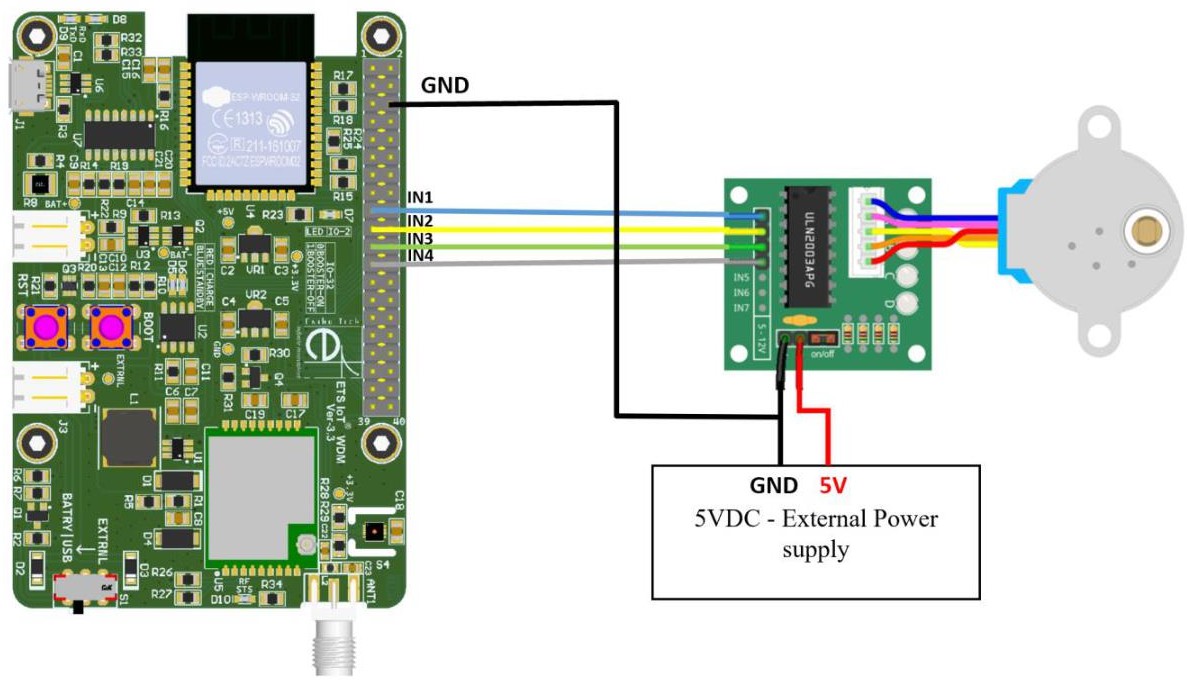
* Given program sketch and AT command should be uploaded before doing the following procedure.
* Open serial monitor console of Arduino IDE and configure the baud rate as 115200 and set line ending as newline.
* The next step is to configure LoRaWAN device with the credentials that obtained from TTN.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

**AT Commands:**

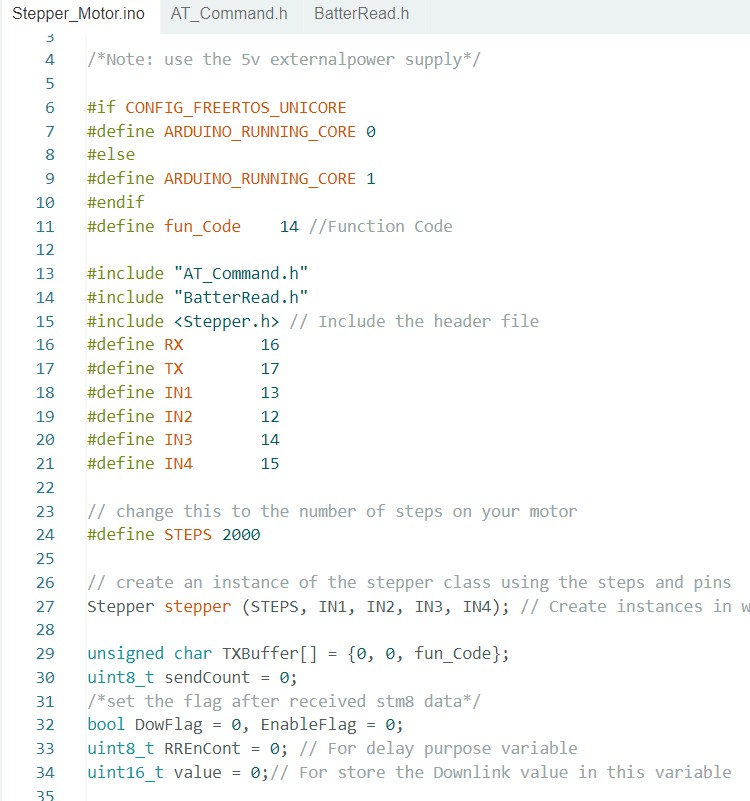
1. AT Commands needed at Initial to store Keys via ESP32 serial port.
   * AT Commands to set initially (Mandatory):

* **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)

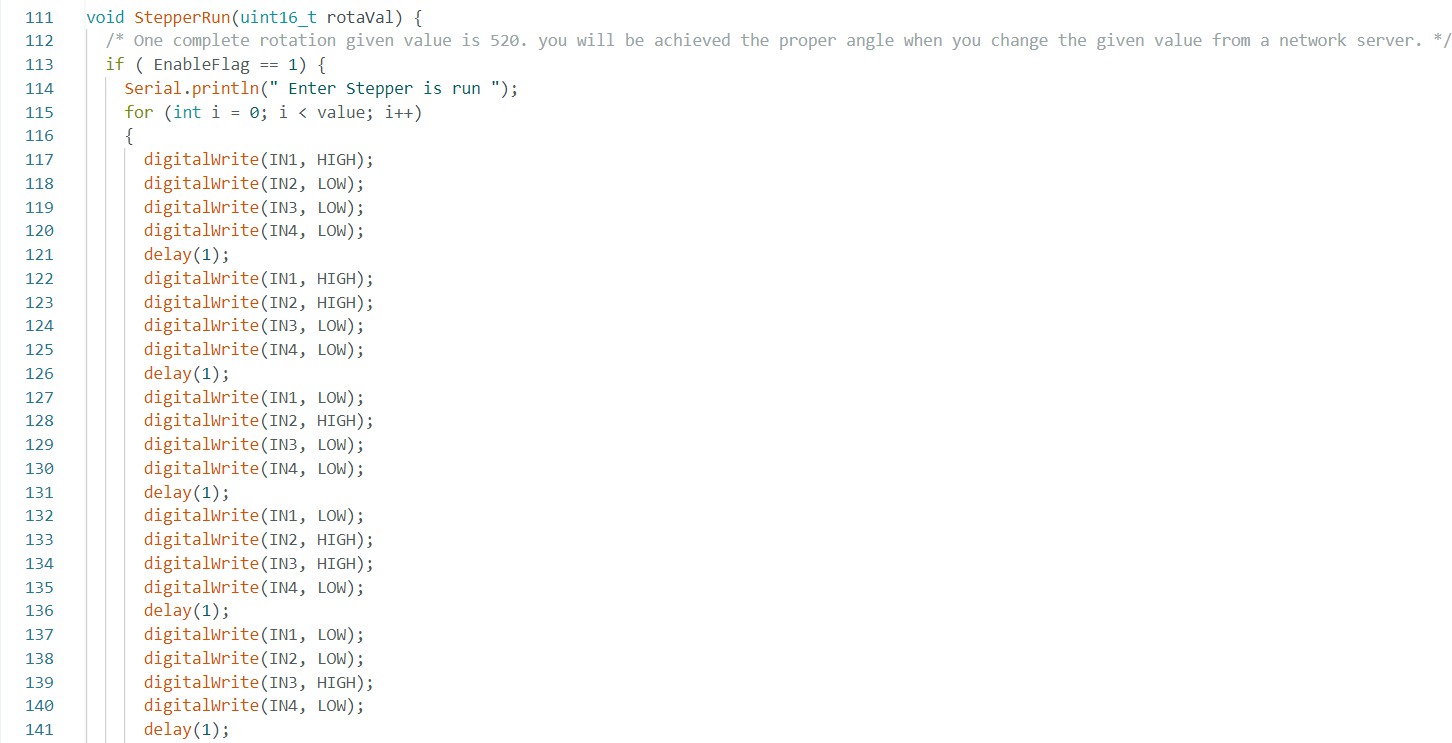
**CIRCUIT DIAGRAM:**

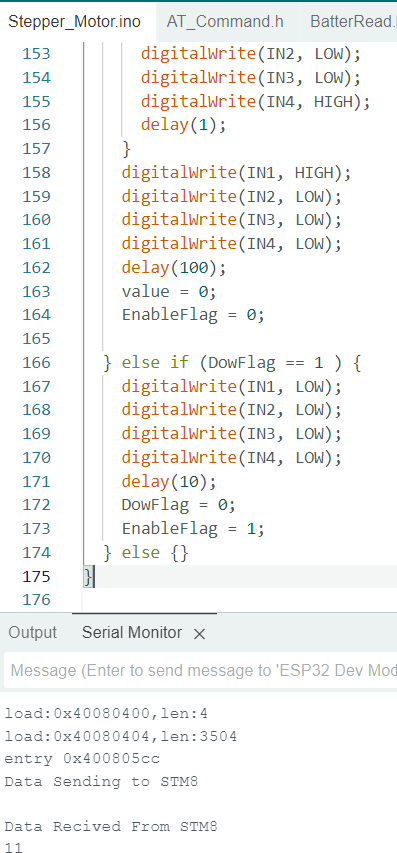
****

**Program:**



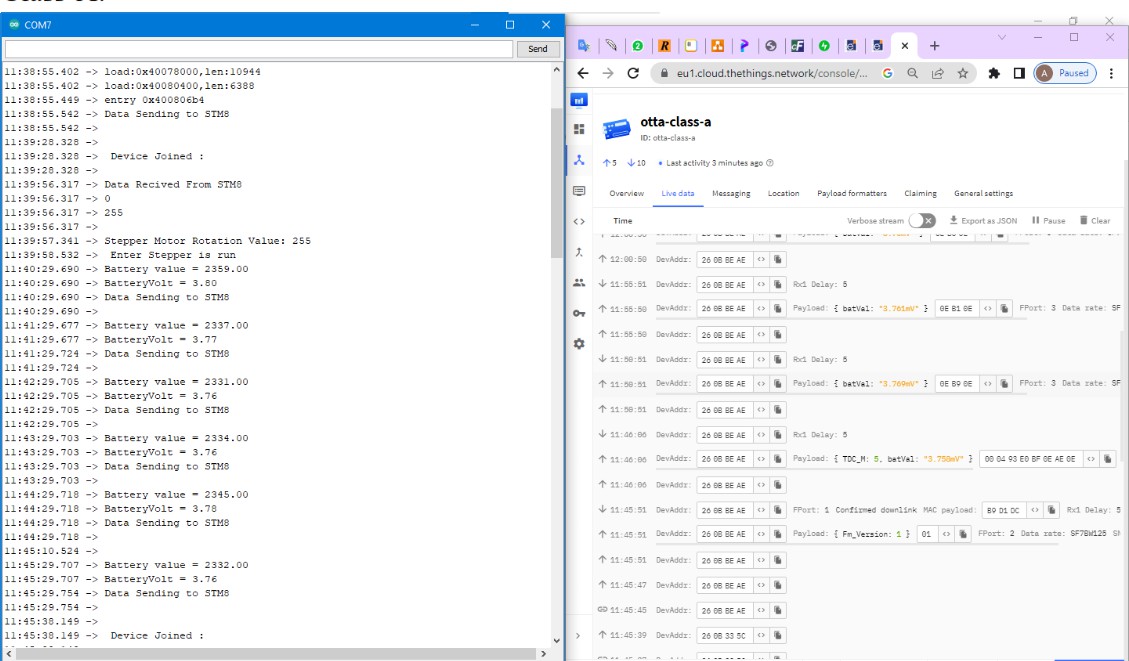


****

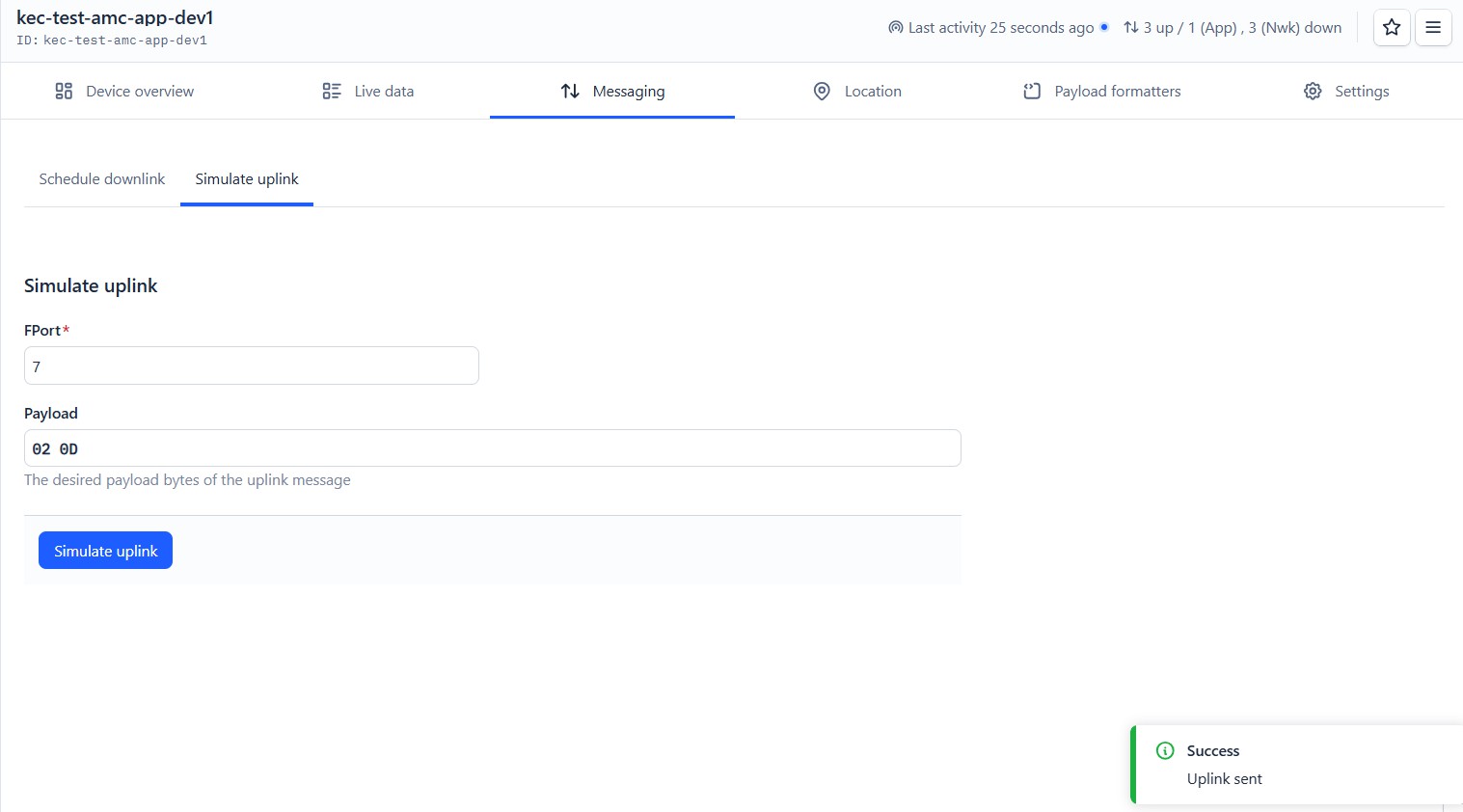


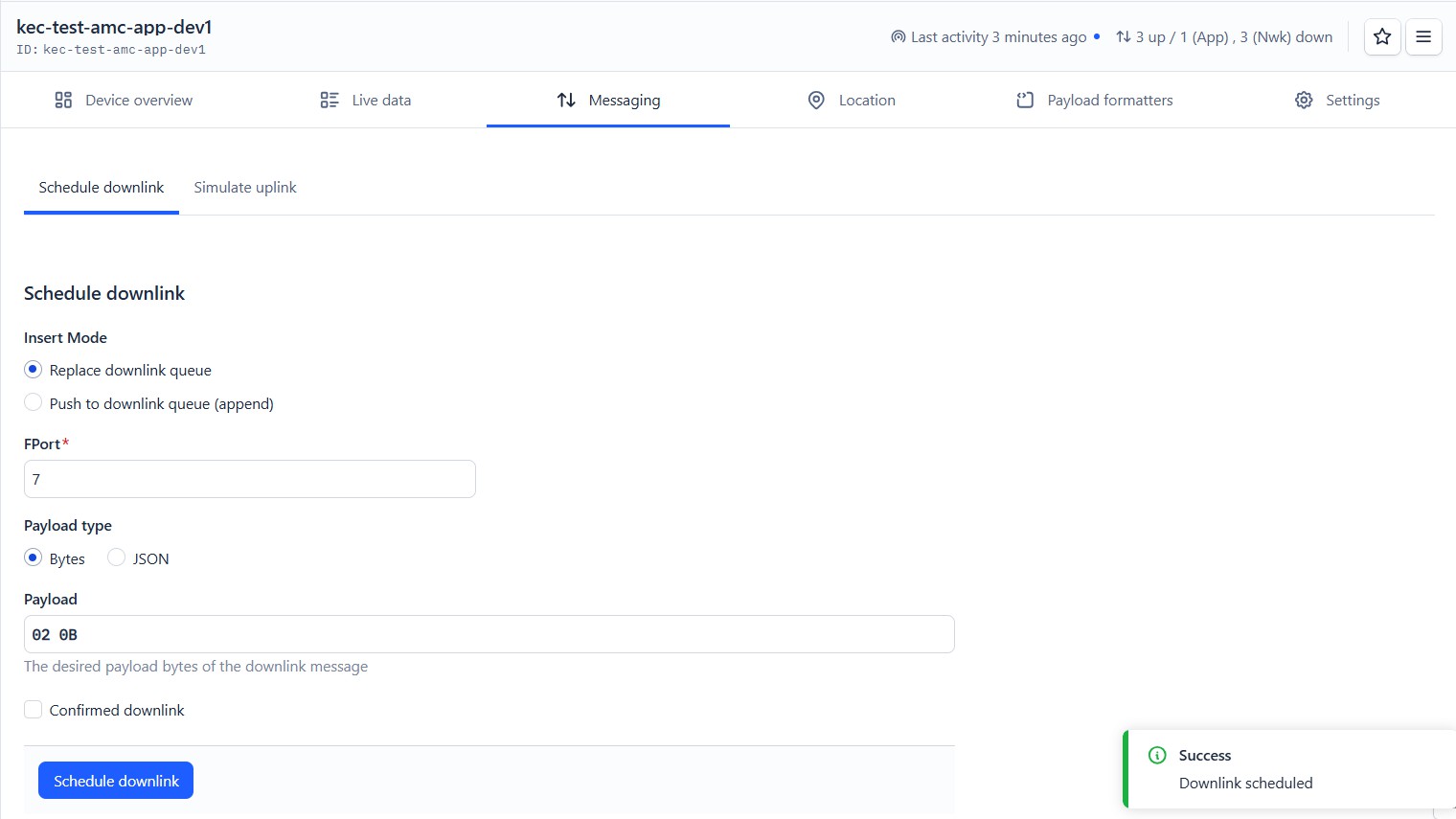
**Simulation output:**

**1. Status Monitoring**

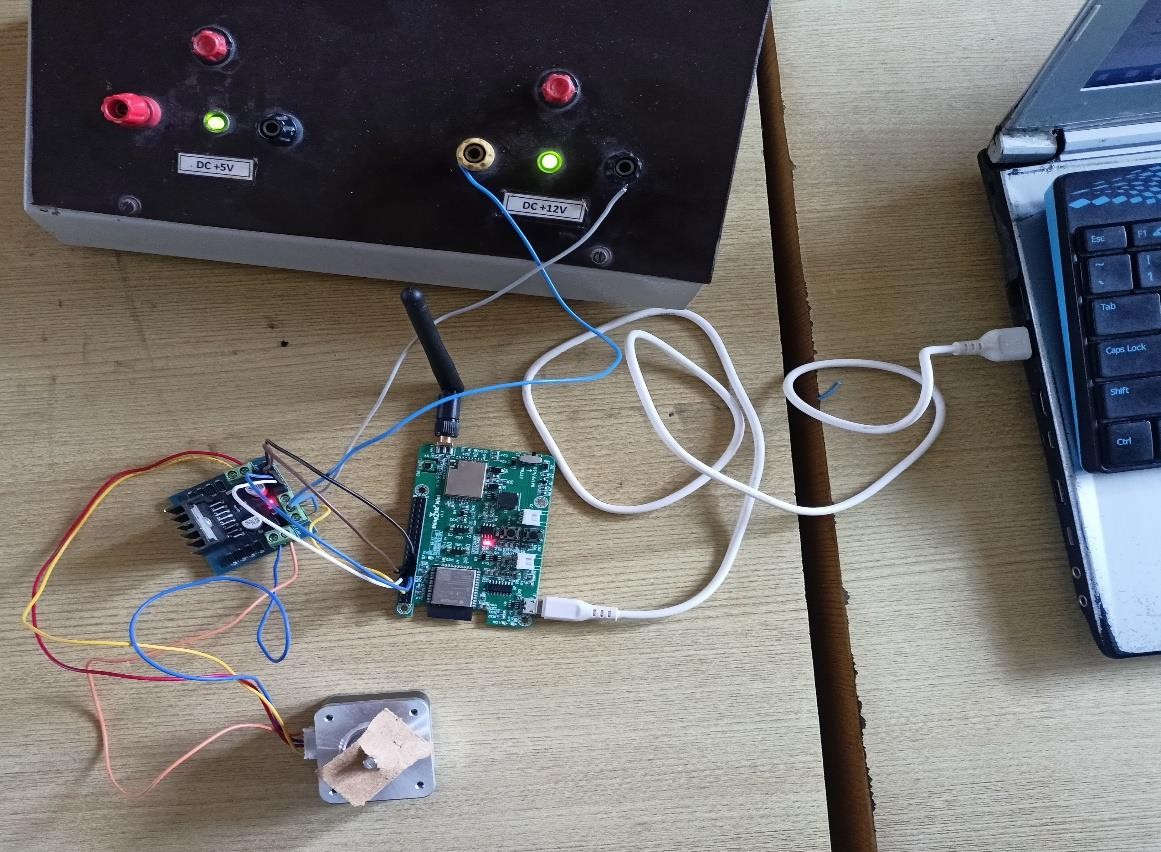
****

1. **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.