LAB PROJECT
ADVANCED EMBEDDED SYSTEMS

WEATHER REPORTING SYSTEM

Patrick NONKI
Elijah Obi
Georges ENEKWA
Abdul-Azeez OLANLOKUN
Guided By - Prof.Dr Ali HAYEK
Department of Electronics engineering
Hochschule Hamm-Lippstadt , Germany

1 INTRODUCTION

The Internet Of Things Helps To Control And Monitor Different Devices Wirelessly Over The Internet. The Internet Acts As A Medium For Communication Between All The Connected Devices. There Are A Large Number Of Applications Of Internet Of Things And Out Of These Applications In This Blog We Will Learn About The Weather Reporting System Using IoT.
For The Efficient Monitoring Of Weather And Other Changes In The Environment, A Weather Reporting System Using IoT Is Quite Necessary. But The Weather Reporting System Should Be A Smart System. The Conventional Weather Reporting System Uses Various Instruments Like Thermometers, Barometers, Rain Gauge Etc. These Instruments Use Analog Technology And Its Data Is Later Stored In A Database Physically. But This System Using Analog Instruments Has Some Limitations.

2 CONCEPT DESCRIPTION

The block diagram we designed to describe our project is in the figure 1 in the appendix.
Weather reporting system: Our system is supposed to get weather conditions, print it on an app via WiFi com-

munication and then based on values we turn on the Green LED and open the windows when the weather is good and turn on the Red LED and close the window when the weather is bad.
We use the following devices, sensors and App for our project:

- **Raspberry Pi**: used as a broker in our project
- **Arduino Uno WiFi Rev.2**: used to receive sensor data and send it with the MQTT Protocol to subscribers
- **KY-015 DHT11 Humidity and Temperature sensor**: used to detect the ambient temperature and humidity
- **Jumping wires**: used to connect the screen and the Sensor to the Arduino respectively
- **Red and Green LEDs**: used respectively to show if the weather is bad or not depending on sensor values
- **LCD 16*2**: used to print out the Temperature and humidity value
- **IoTMQTTPanel App**: used to receive sensor data via MQTT and plot temperature and humidity graphs as well as printing text values.

3 PROJECT/TEAM MANAGEMENT

The project management method used was agile methodology, specifically SCRUM, where we had weekly sprint sessions(physically) and then after review of done tasks, the milestones were adapted regarding to what has been done and what has to be done. Complex tasks were broken down to smaller ones and assigned to individuals.

- The **MQTT Setup and Raspberry Pi OS installation** was done by **Patrick NONKI**. He took care of making sure that the Mosquitto broker is installed on the Raspberry Pi and that the communication with the publishers and subscribers is possible.
- **The Arduino Code including the publishing message** were done collaboratively between **Patrick NONKI and Elijah OBI**. The ensured that the data from the sensor is well retrieved and correctly transmitted to the subscriber via MQTT.
- **The Physical Implementation on the breadboard** was done by **Georges ENEKWA and Abdul-Azeez OLANLOKUN**. The ensured according to the Datsheets that the sensor and the screen are connected to the Arduino WiFi and correctly working.
- The IOTMQTTPanelApp was done by **Georges ENEKWA and Abdul-Azeez OLANLOKUN**. They ensured that the data from the Arduino is correctly received by the App.

4 TECHNOLOGIES

This section is to describe what and how are the technologies we used in our project.

- **Arduino Uno WiFi Rev.2**: The Arduino Uno WiFi Rev2 is an Arduino Uno with an integrated WiFi module. The board is based on the Microchip MEGA4809 with an ESP32 u-blox NINA-W13 WiFi Module integrated. The NINA-W13 Module is a self contained SoC with integrated TCP/IP protocol stack that can give access to your WiFi network (or the device can act as an access point).
  The Arduino Uno WiFi Rev2 is programmed using the Arduino Software (IDE), an integrated Development Environment common to all the boards and running both online and offline.
- **DHT11**: The DHT11 is a commonly used Temperature and humidity sensor that comes with a dedicated NTC to measure temperature and

an 8-bit microcontroller to output the values of temperature and humidity as serial data.
It has the following specifications:

  - Operating Voltage: 3.5V to 5.5V
  - Operating current: 0.3mA (measuring) 60uA (standby)
  - Output: Serial data
  - Temperature Range: 0°C to 50°C
  - Humidity Range: 20% to 90%
  - Resolution: Temperature and Humidity both are 16-bit
  - Accuracy: ±1°C and ±1%

- **Raspberry Pi model 3b** : Raspberry Pi 3 Model B is the earliest model of the third-generation.It has the following specifications:

  - Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
  - 1GB RAM
  - BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
  - 100 Base Ethernet
  - 40-pin extended GPIO
  - 4 USB 2 ports
  - 4 Pole stereo output and composite video port
  - Full size HDMI
  - CSI camera port for connecting a Raspberry Pi camera
  - DSI display port for connecting a Raspberry Pi touchscreen display
  - Micro SD port for loading your operating system and storing data
  - Upgraded switched Micro USB power source up to 2.5A

- **LCD 16*2** : The 16×2 liquid crystal display can be used in embedded systems where displaying a limited amount of data is necessary. This display comes with two lines of data and each line is divided into sixteen columns. Each row has a block of 8 rows and 5 columns also called cells or in other words we can say that each cell of the row has 40 pixels.
  The configuration of the device will be shown in the arduino code.

- **IoTMQTTPanel App**: This application allow you to mange and visualize IoT project, based on MQTT protocol.

Among that the most important part is the communication protocol between the Arduino WiFi, the Raspberry Pi and the IoTMQTTPanel App. We decided to work with the MQTT Protocol for this project. MQTT is an OASIS standard messaging protocol for the Internet of Things (IoT). It is designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth.

5 IMPLEMENTATION

To be able to achieve our project, we followed the following steps to be sure that everything is working fine.

5.1 Setting up the Raspberry Pi and Installing the Broker

- The first step is to download the operating system that will be installed in the SD Card and plugged into the Raspberry Pi. For that, go to the link and click on manually install an image, then the Raspberry Pi Os Lite.
- Now you will need an imager to flash your SD Card with the OS you downloaded.
- Then you will plug the SD Card in the Raspberry Pi, plug an ethernet cable, power on your device and connect it to a screen via HDMI.

- The next step will be to install the Mosquitto broker on your device for it to be able to work as a broker for the MQTT Protocol.
  The first step is to update your OS with the following command lines :

```
$ sudo apt update
$ sudo apt upgrade
```

  Now you type the following lines of codes to install the mosquitto broker:

```
$ sudo apt install mosquitto mosquitto-clients
```

  To be sure now that the mosquitto is installed and active, we will have to type the following line of code and get the response *(active) running* :

```
$ sudo systemctl status mosquitto
```

- Let's now fix a static IP address to our raspberry Pi so that it always has a similar IP address independently of the network.
  - First you will have to go to the dhcpcd.conf file to edit it. Type the following command :

```
$ sudo nano /etc/dhcpcd.conf
```

    The file will open and look like this :

```
interface NETWORK
static ip_address=STATIC_IP/24
static routers=ROUTER_IP
static domain_name_servers=DNS_IP
```

  - Replace the names as follows :
    - Replace NETWORK by your connection type (eth0 or wlan0)
    - Replace STATIC-IP by the IP you want for your Raspberry Pi
    We can have the following example :

```
interface wlan0
static ip_address=192.168.1.120/24
static routers=192.168.1.254
static domain_name_servers=192.168.1.254
```

  - Once you have entered the settings, press **CTRL + X** and then **Y** and **ENTER** to save the modified file.
  - Let's reboot now the Raspberry Pi with the following command to apply all the changes:

```
$ sudo reboot
```

    The IP Address is now set to a static value and will be the same on every network.

5.2 The Arduino Code

Let's describe now the Arduino Code used as the publisher.

```
#include <dht.h>
#include <LiquidCrystal.h>
#include <ArduinoMqttClient.h>
#include <WiFiNINA.h>
#include <SPI.h>
#include "arduino_secrets.h"
#define dht_apin A0 // Analog Pin sensor is connected to
```

Here, we define all the libraries that we'll use in our project and we define the A0 pin to be connected to the DHT data pin.

```
const char broker[] = "192.168.2.148";
int        port     = 1883;
const char topic[]  = "Sensor";
const char topic2[]  = "Temperature";
const char topic3[]  = "Humidity";
```

Now we define the IP address of the broker (Raspberry Pi), the port and the topics on which the subscriber will subscribe to have access to the data from the Arduino.

```
DHT.read11(dht_apin);

 if (DHT.temperature>25){
   digitalWrite(10,HIGH);
   delay(2000);
   digitalWrite(10,LOW);
 }else{
   digitalWrite(7,HIGH);
   delay(2000);
```

```
10      digitalWrite(7,LOW);
11    }
```

Now inside the loop we get the values from the DHT and turn on the green LED or Red LED if the temperature is greater than 25 or not.

```
1 mqttClient.beginMessage(topic);
2   mqttClient.print("The␣temperature␣
        is:␣");
3   mqttClient.print(DHT.temperature);
4   mqttClient.print("C␣and␣the␣
        humidity␣is:␣");
5   mqttClient.print(DHT.humidity);
6   mqttClient.print("%");
7   mqttClient.endMessage();
```

This part here is to send data via MQTT to the broker with the topic specified at the beginning.
The final and complete code can be found in our github.

5.3 The Hardware Configuration

- For the LCD screen we'll just follow the rules of the figure 2.
- For the DHT 11, the pin with the mark *s* is connected to the A0 of the Arduino and the other pins are connected to the 5V and the GND.

APPENDIX

5.4 The MQTT App

- First we have to add a connection by editing the name of the connection, the IP address, the port and the network protocol according to the figure 3.
- Then we have to choose a panel from the list. See figure 4
- Then we create a text log where we'll receive our data from the publisher (Arduino). See figure 5.

6 USE CASE

To be able to use our project, nothing specifically has to be done. Just power on your raspberry pi and your Arduino and then subscribe to the topics on your App and you'll be able to retrieve the ambient temperature and humidity each second.

REFERENCES

[1] Pr.Dr Ali HAYEK (2022) *Advanced embedded systems Lab course*, Hochschule Hamm-Lippstadt.

[2] Utkarsh Pathak, *Weather Reporting System Using IOT*, Pianalytix Edutech Pvt Ltd, 2020.

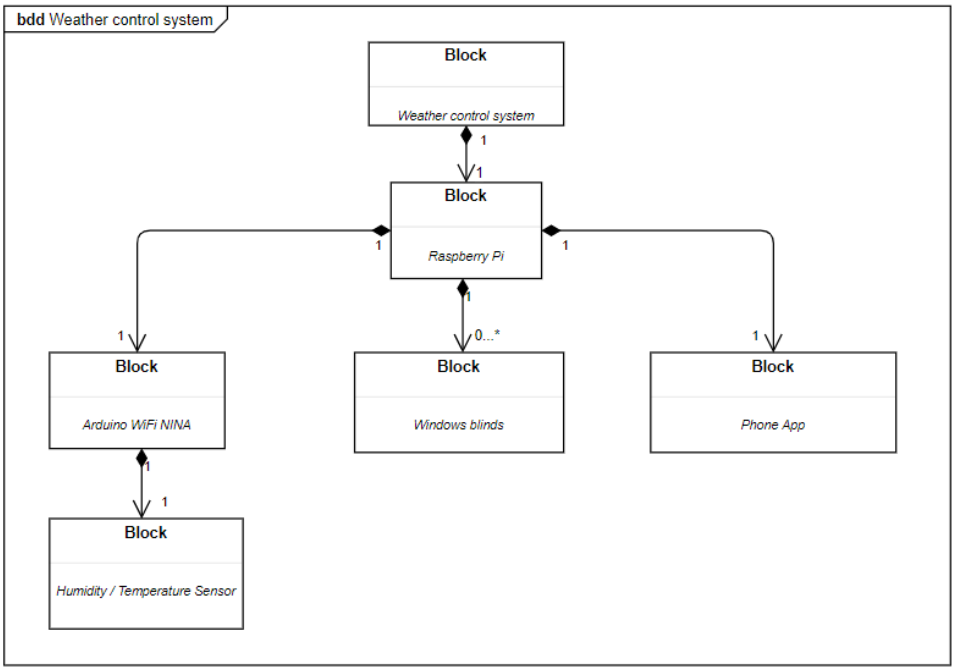

Figure 1: Block diagram

To wire your LCD screen to your board, connect the following pins:

- LCD RS pin to digital pin 12
- LCD Enable pin to digital pin 11
- LCD D4 pin to digital pin 5
- LCD D5 pin to digital pin 4
- LCD D6 pin to digital pin 3
- LCD D7 pin to digital pin 2
- LCD R/W pin to GND
- LCD VSS pin to GND
- LCD VCC pin to 5V
- LCD LED+ to 5V through a 220 ohm resistor
- LCD LED- to GND

Additionally, wire a 10k potentiometer to +5V and GND, with it's wiper (output) to LCD screens VO pin (pin3).

Figure 2: LCD

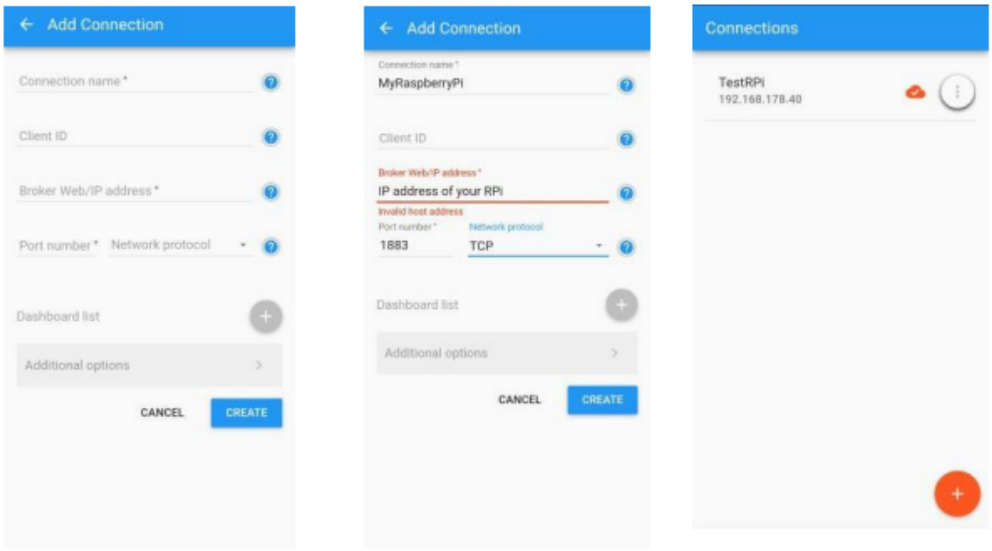

Figure 3: Connection

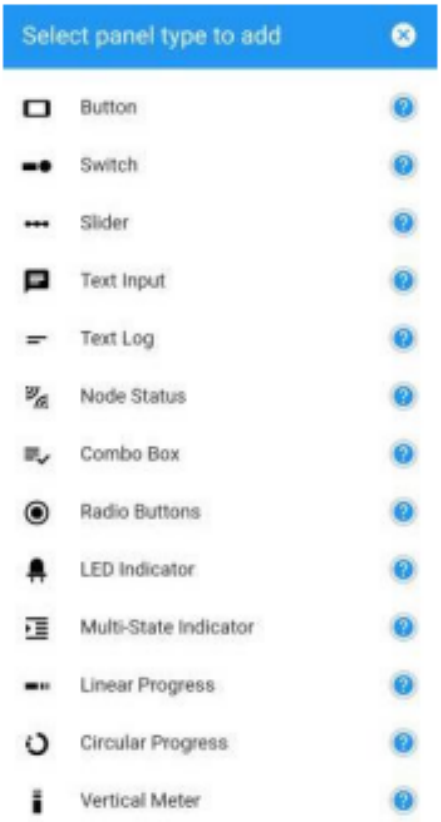

Figure 4: Panel

Figure 5: Text log