Milestone 2

10 November 2024

Deadline: Saturday 30 November 11:59 pm

Important Note: Read the whole document very well before you start implementing or buying any hardware. Read the deliverables and submission guide well to know what you will be submitting and how.

In this milestone you will be working in teams of 3

1 Introduction

Using Internet of Things (IOT), we can control any electronic equipment in homes and industries. Moreover, you can read a data from any sensor and analyze it from anywhere in the world.
Assume you have an IoT sensor (ex: temperature, light, RFID...) connected to the analogue input of an ESP8266 Wi-Fi module which will periodically send the readings to the server, using client-Server model and socket programming.

In this Milestone, you are required to build a smart system. The system should include 2 sensors connected to 2 ESP8266 Wi-Fi modules, a central server to receive and process the data from the sensors. The server should send a feedback to the ESP8266. If the data is within the normal range, continue sending the data. However, if readings were out of range a warning sign should arise.

2 Hardware Requirements

1. 2x ESP8266 modules: Make sure that the modules are connected to a board that has USB interface to be programmed through it and pins. Refer to Figure 1

Figure 1: ESP8266 on a board

2. 2x DHT11 Temperature and Humidity Sensors

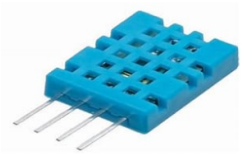

Figure 2: DHT11 Sensor

3. 2x LED Lamps
4. 2x 220 Ohm Resistors
5. Jumpers: You'll need both male to male and male to female jumpers.
6. Breadboard

3 Software Requirements

In this Milestone you are required to program the ESP8266 microchip. To do this you will be using micropython language. You'll first need to download an IDE to write and upload the code. Then upload the micropython firmware to the esp8266 module. Here are some steps that may help you.

1. Download and install Thonny IDE from Here
2. Download MicroPython firmware for ESP8266 from Here
3. Flash the firmware to your Esp8266 using Thonny
   (a) Connect your ESP8266 board to your computer.
   (b) Open Thonny IDE. Go to Tools > Options > Interpreter.
   (c) Select the interpreter for ESP8266 and select the COM port your board is connected to. Finally, click on the link Install or update firmware.
   (d) Select the port once again, and then click on the 3 dash button at the bottom right > Select local micropython image. Now browse to the .bin file with the firmware you've downloaded. Finally click on Install.

For more detailed guide check this Tutorial.

4 Implementation

The overall goal is to make an alarm by lighting the LED lamps whenever the sensors readings are out of "normal" range. The ESP8266 should get the readings from the sensors, one for each sensor, and send the readings to a server (your laptop). The server should check the readings and notify the clients (ESP8266) if the readings are not within a specific range. The clients then should light their corresponding LED lamps until temperatures are back to normal. Here's what you should do:

1. Prepare the client code and upload it to both esps. The code is written in micropython.
   (a) The code should get the humidity and temperature readings from the sensor and send it to the server periodically and non-stop.
   (b) The code should connect to the server using wi-fi and TCP sockets as in the previous milestone.
   (c) The client should send the sensor readings (temperature and humidity) to the server.

(d) If the client receives a response from the server indicating abnormal temperatures, the esp client should turn the LED lamp on until it gets another update from the server.

(e) You should set an appropriate timer between each reading.

2. Prepare the server code in python.

   (a) The server should apply threading to support the two clients' connections.

   (b) The server should receive the data from the sensors and check if it is within the normal range.

   (c) The server should send messages to the clients indicating normal or abnormal readings.

   (d) You should choose an appropriate range of temperatures to be "normal".

3. Both sensor readings should be treated separately. Ex: If sensor 1 sends normal temperature but sensor 2 readings were out of range, only LED 2 should turn on.

4. Connect the circuit on your breadboard. Before you power it on, double-check all the connections to avoid damaging any parts and to save your money for a treat!

   Tip 1: Use the board as a power source and ground for your circuit.
   Tip 2: The resistors are used in series with the LED lamps to limit the current so it won't burn.

5. Test the functionality of the project with different scenarios for readings, ex: (normal, normal), (abnormal, normal), (abnormal, abnormal).

6. **BONUS STEP**: Adjust your project to give indication if the temperature is too high or too low.

   (a) Change in server code to indicate if temperature is too high or too low instead of just normal or abnormal

   (b) Change in client code to light two different LEDs based on the scenario

   (c) Add another LED (with different color) to one of your esps so, for example, if temperature is too low the yellow LED turns on and if the temperature is too high, the red LED goes on.

5 Deliverables and Submission Guide

Deadline: Saturday 30 November at 11:59 pm

In this milestone you will be working in teams of min 2 and max 3.

5.1 Deliverables

5.1.1 Report

Every team should prepare a report discussing the project and their own implementation details. You should include the following:

1. Name of the team, name and ids of the members
2. Introduction and goal of project
3. Code shots with comments and descriptions discussing the functionality.
4. Discuss your choices of timer and temperature ranges.
5. Diagram of the circuit connection (Try to make it professional not a real picture of the circuit!)
6. Step by step guide to connections you made, including pins, input, output, voltage source, ... etc.
7. Discuss the implementation and test cases you did supported by shots, photos, diagrams, ... etc.
8. Discuss how you tested your project
9. Discuss possible use-cases for this project.

5.1.2 Video

Every team should record a small video of the circuit in action. The video should show the following:

1. How you test your circuit and change the temperatures
2. The readings of the sensors at client or server side
3. The reaction of LEDs to the various temperatures

5.1.3 Codes

Each team should also submit the client and server codes.

5.2 Submission Guide

For the submission of this milestone, each team should:

1. Create a new drive named by the team name.
2. Upload all your deliverables (report, video, code) to this drive
3. Send your drive link through this form. You can fill this form now so you won't forget later.

VERY IMPORTANT NOTE 1: TRIPLE-CHECK THAT THE LINK TO THE DRIVE IS ACCESSIBLE. NO ACCESS = ZERO!

IMPORTANT NOTE 2: Make sure the team name is the same in the report, drive name and google form, otherwise = zero.

IMPORTANT NOTE 3: I will not accept submissions by mail.

IMPORTANT NOTE 4: Needless to say, cheating case = zero.

GOODLUCK!