UNIVERSITY OF CALIFORNIA, DAVIS
Department of Electrical and Computer Engineering
EEC 172 Winter 2022

LAB 5 Verification

Team Member 1: Abraham Hedaf

Team Member 2: Sampson Ezieme

Section Number/TA: A03

Demonstrate your working application to your TA

Date	TA Signature	Notes
4 MAR 2022	Alex	Completed the final project that they set out to accomplish. Heat sensor consistently works, email notification system works, and fan circuit works.

CC3200 Fire Protection Fan System

Abraham Hadaf & Sampson Ezieme
EEC 172

1. INTRODUCTION

The purpose of this lab was to use the REST API to connect to a web service and sense the temperature of the board and turn on a fan when the board gets too hot, in order to cool it down, and to send a notification via email to let the user know that the board has reached a dangerous level. Afterwards, a DC motor fan connected to the CC3200 cools down the area and sends a warning notification to Amazon AWS email notification.

2. BACKGROUND

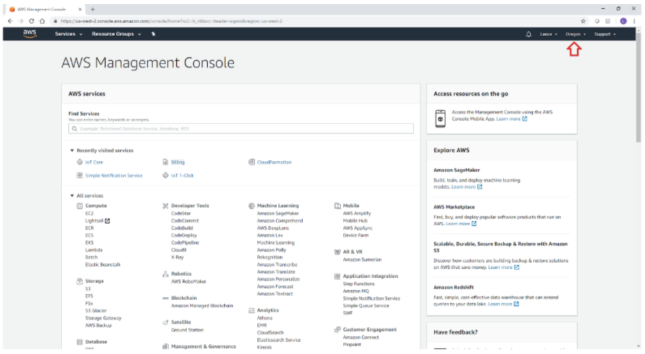
2.1 REST API

The RESTful API specifies the format of messages sent to a service to interact with it. The RESTful APIs' functionality varies depending on the service with which you're interacting. The following are some of the most used commands. The RESTful API is a web-based communication protocol based on the Representational state transfer (REST) architecture that is widely used for many services. HTTP is commonly used to implement RESTful APIs. RESTful API is widely used due to its simplicity, despite not being as optimized for low power and low-cost applications as another IoT protocol, MQTT.

2.2 AMAZON WEB SERVICES

Amazon Web Services (AWS) is a collection of cloud-based web services that eliminates the need for individual organizations to construct the hardware infrastructure for these efforts. The Internet of Things (IoT) service was recently released, allowing numerous physical products to be connected.

AWS IoT uses the concept of a device shadow to represent these real-world devices in the cloud and keep them in a persistent/coherent state despite intermittent network connections. The condition of the real-world gadget in the cloud is represented by this shadow. The device transmits its modifications to the shadow device on AWS through MQTT or a RESTful API when it changes states. If the device loses network connection and then reconnects, it can update itself by pulling the last-known state from the shadow. Updates to this shadow can also use Rules to trigger actions in other services. A rule is utilized in this lab to send an email to your email address.



2.3 TEMPERATURE SENSOR AND OLED DISPLAY

The LaunchPad CC3200 manufactured by Texas, is equipped with two sensors that use I2C communication, an accelerometer and a temperature sensor. The sensor's address is 0x41 for the temperature sensor and 0x18 for the accelerometer. By utilizing the 0x41 address, we are able to initialize the temperature sensor and detect the temperature of the CC3200 environment. Temperature Sensors measure the amount of heat energy or even coldness that is generated by an object or system, allowing us to “sense” or detect any physical change to that temperature producing either an analogue or digital output.

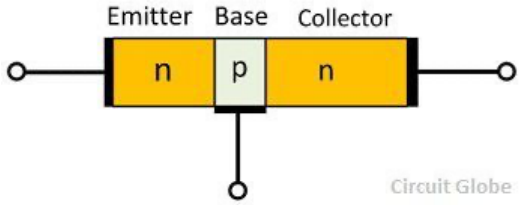
We used SPI to connect the CC3200 LaunchPad to the 128x128 color organic light-emitting diode (OLED) display. The serial peripheral interface (SPI) is a popular interface between microcontrollers and peripheral ICs like sensors, ADCs, DACs, shift registers, and SRAM. SPI is a full-duplex, synchronous master-slave interface. On the rising or falling clock edge, data from the master or slave is synchronized. Data can be transmitted by both the master and the slave at the same time. In short, in this communication protocol, devices exchange data in master/slave mode. The master device is mainly responsible for the initiation of the data frame. The master device also selects the slave device to which data need to be transferred.

2.4 DC MOTOR AND NPN TRANSISTOR

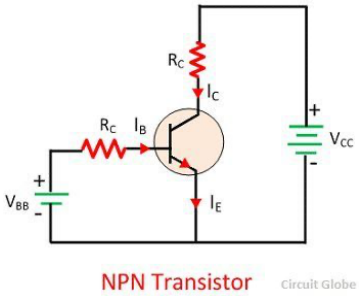
NPN transistors are devices in which one p-type material is placed between two n-type materials. The NPN transistor amplifies weak signals at the base and generates strong signals at the collector end. The movement of an electron in an NPN transistor is from the emitter to the collector region, resulting in current in the transistor. Because the majority of charge carriers in such transistors are electrons, which have a higher mobility than holes, they are commonly used in circuits.

The NPN transistor has two diodes connected back to back. The diode on the left side is called an emitter-base diode, and the

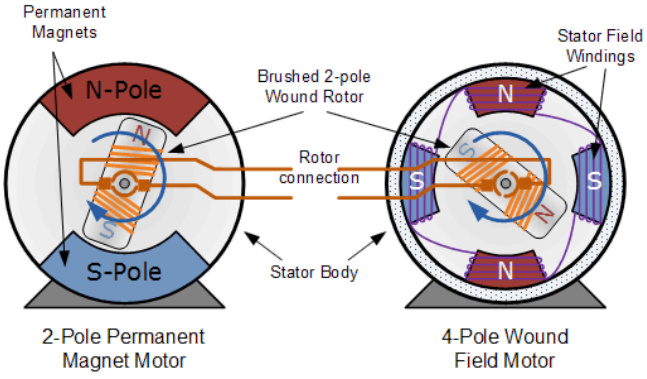
diodes on the left side are called collector-base diode. These names are given as per the name of the terminals.



The NPN transistor's circuit diagram is shown in the diagram below. In reverse bias, the collector and base circuits are connected, while the emitter and base circuits are connected in forward bias. For controlling the ON/OFF states of the transistor, the collector is always connected to the positive supply, while the base is always connected to the negative supply.



The DC Motor, or Direct Current Motor to give it its full name, is the most commonly used actuator for producing continuous movement and whose rotational speed can be easily controlled, making it ideal for applications requiring speed control, servo type control, and/or positioning. A DC motor is made up of two parts: a "Stator," which is stationary, and a "Rotor," which rotates.



Normal DC motors have almost linear characteristics, with the applied DC voltage determining the speed of rotation and the current flowing through the motor windings determining the output torque. Any DC motor can rotate at a speed ranging from a few revolutions per minute (rpm) to many thousands of rpm, making it suitable for electronic, automotive, or robotic applications. By connecting them to gearboxes or geartrains, their output speed can be reduced while the torque output of the motor can be increased at a high speed.

The text should be in two 8.45 cm (3.33") columns with a .83 cm (.33") gutter.

3. GOALS

3.1 Sensing the Room Temperature

The goal is to use the onboard temperature sensor of the CC3200 and, with I2C communication through address 0x41, use the communicated temperature values to run some functionalities. Specifically, when the sensed temperature reaches more than 80 degrees Fahrenheit.

3.2 Connecting Temperature Sensor and Fan to AWS

The first goal is to connect the CC3200 to AWS. There are many steps to achieve this. AWS uses the concept of a device shadow to represent your thing (i.e. your CC3200 board) in the cloud, from which you can retrieve the most recent state or push a new state to the server. As a result, even if your device is only connected to a wireless network intermittently, it can pull the most recent state from the cloud and maintain continuity. On AWS, however, appropriate security measures can be implemented to ensure that the devices have the authority to carry out the requested actions.

To do this you must perform the following:

- Create a thing for the cc3200 as you have done before.

- Create an SNS topic

- Subscribe to the SNS topic with your email. Create a rule to forward incoming thing messages to your SNS topic. Use the REST API to post to your topic.

If the temperature sensor detects a temperature that's too hot, it will send an email notification that warns the user that CC3200 is too hot.

Not only that, the DC Motor fan should turn on to cool the environment when the temperature is too hot. The DC motor will be controlled by an NPN transistor which will behave like a switch for the DC Motor.

3.3 Syncing Temperature Sensor to OLED

To sync the temperature to the OLED, the goal is to connect the OLED to the CC3200 using SPI communication. WriteData() and WriteCommand() are low-level functions that use the SPI port to write a data or command byte to the OLED. In the beginning of the lab, we looked at the SPI example program and put it to the test. You can use the higher-level OLED library functions with the WriteData() and WriteCommand() functions. We used the interface signals listed in the table in the lab manual 3 for the SPI interface to the OLED. To control the DC (Data/Command), OLEDCS (OC), and RESET signals on the OLED, we used GPIO signals configured as outputs and connected the pins accordingly to the CC3200 and the OLED.

The OLED will display to the user that the Safety is on" which means that the temperature is not hot and the environment has a stable temperature. However if the temperature is too hot, the OLED will display "Danger! The CC3200 is too hot!".

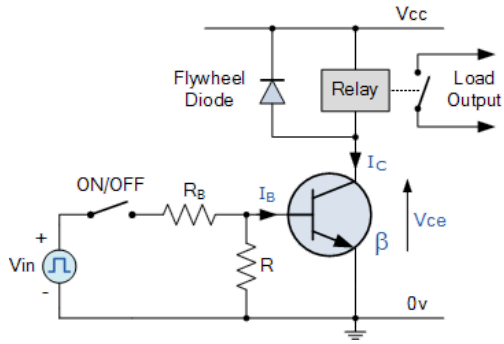
4. METHODS

4.1 Using the NPN Transistor as a Switch for the DC Motor

By biasing the transistor's Base terminal differently than for a signal amplifier, NPN type bipolar transistors can be made to operate as "ON/OFF" type solid-state switches.

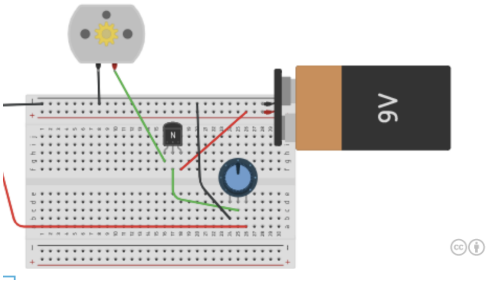
One of the most common uses for using a transistor to turn a DC output "ON" or "OFF" is solid-state switches.

To use the transistor as a switch, it must be either completely "OFF" (cut-off) or completely "ON" (on) (saturated). When turned "fully-OFF," an ideal transistor switch would have infinite circuit resistance between the Collector and Emitter, resulting in zero current flow, and when turned "fully-ON," it would have zero resistance between the Collector and Emitter, resulting in maximum current flow.



Modest leakage currents flow through the transistor while it is turned "OFF," and when it is fully "ON," the device has a low resistance value, creating a small saturation voltage (VCE) across it. The transistor dissipates the least amount of power in both the cut-off and saturation areas, despite the fact that it is not a perfect switch. The Base input terminal must be made more positive than the Emitter by elevating it over the 0.7 volts required for a silicon device in order for the Base current to flow. The Base current is controlled by adjusting the Base-Emitter voltage VBE, which influences the amount of Collector current passing through the transistor as previously explained. The transistor is considered to be saturated when the Collector current reaches its maximum value. The value of the Base resistor controls how much input voltage and Base current are required to fully "ON" the transistor.

We set the transistor GPIO to “high” when the temperature sensor detected a high temperature and “low” when the temperature sensor was detecting a cool temperature. When the transistor was set to high, the DC motor was able to turn on as expected.



We used a potentiometer to control the speed of the DC Motor. Although inefficient, it was quite effective and simple to implement.

4.2 Communicate OLED through SPI

For successful communication between the OLED and CC3200, the WriteData() and WriteCommand() are used in the Adafruit_OLED.c. We designed our code in steps for the WriteData and WriteCommand functions. For the WriteData, we set variable dummy, set DC to high, OLED pin to low, enable chip select, push character of SPI, clean SPI, and then pull OLED to high. For the WriteCommand, it was a similar approach except set DC pin to low. With these steps, we were able to successfully create communication between the OLED and CC3200. After getting the communication working, we used the test.c to create the Danger and Safety functions. We called those functions into main.c and got the OLED working.

4.3 Connect to AWS

First, one must create a device/thing in the IoT section on Amazon AWS. Then, receive the certifications and keys to flash into the CC3200. For the CC3200 to connect to the AWS server, one must

use uniflash to flash the certificate, root, and private key for the CC3200 to properly connect to the AWS server. Since the certifications and keys have an incompatible extension with the CC3200, it must be converted to a .der file which is possible to do using OpenSSL. Finally, one must create a policy to allow Update/Get Status from the Thing Shadows. To demonstrate you can make a POST request, one must change the common.h file to match the WIFI connection and password you are using. Verifying the pins in the pinmux.c and using your own endpoint in the main.c file. To set up your IoT rule, a query statement using SQL is needed to push updates in the CC3200.For this case, I used $aws/things/thingName/shadow/update/accepted as the statement. However, this statement would leave the JSON tags present. I modified the statement to SELECT VALUE, which means the query would extract the value of the data.

When the temperature sensor detects a temperature that’s too hot, an email notification is sent that warns the user that CC3200 is too hot along with the OLED displaying a “Danger! The CC3200 is too hot!” and the DC motor fan turning on.

5. DISCUSSION

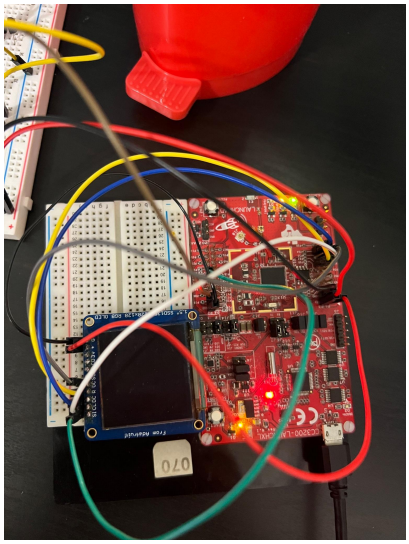
There were some challenges involved with this lab. The first challenge was involved with AWS. Since we couldn’t use our Lab 2 certificates we had to start over from scratch. We ended up using the AWS Root CA and everything else seemed to fall into place after that. Another challenge was being able to test the project’s danger setting. Because the danger region was about 80 degrees

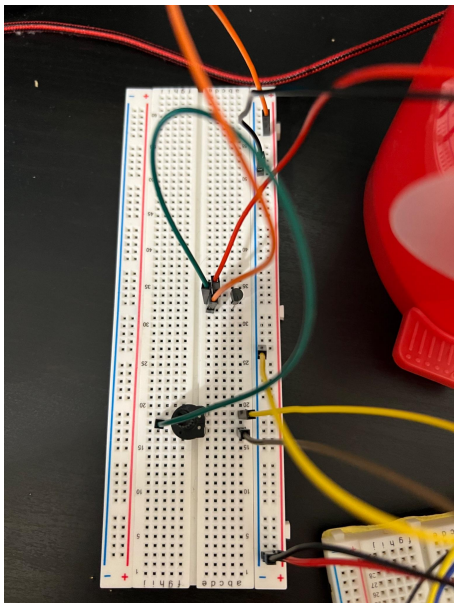
Fahrenheit we couldn't simply reach the threshold because we were bound by the room temperature. To solve this, we would lower the danger threshold temperature to test the AWS communication or we would simply breathe on the sensor and that would sometimes get the temperature above 80 degrees Fahrenheit. The last challenge that we ran into was figuring out how we will be able to add a DC Motor fan to our project in order to cool down the system when the temperature reaches above 80 degrees. We tried just normally connecting the fan to the voltage and ground, but that obviously caused the fan to stay on the entire time, regardless of the temperature. Knowing that this would be very energy inefficient, we decided to implement our DC Motor fan using a switch that would only turn on when we reached the danger threshold of over 80 degrees Fahrenheit. We implemented this using an NPN-type bipolar transistor and once we did everything started to work, but not as intended. The fan would run at the max speed so in order to control the speed of the fan to something more comfortable we implemented a potentiometer to increase or decrease the resistance and get a more desirable fan speed. After this was implemented, everything started to run as we intended it to.

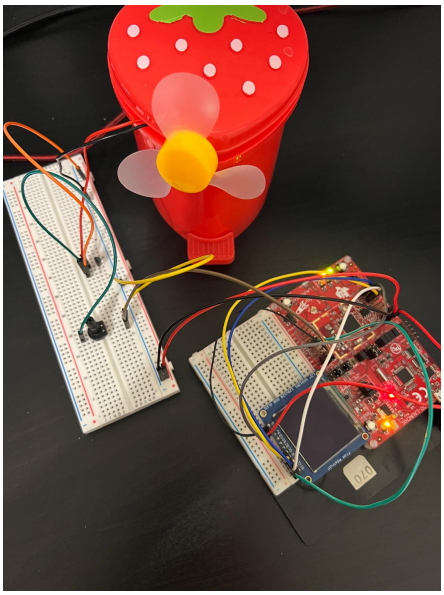
6. CONCLUSION

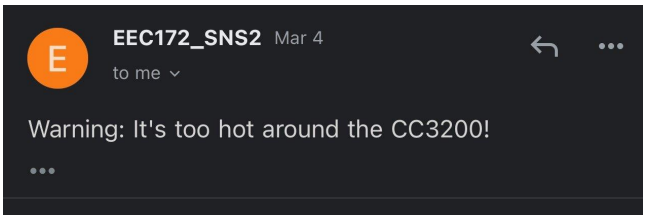
Throughout this lab, we learned how to better use and implement the onboard sensors, REST API, AWS, DC Motors, transistors, and potentiometers. We had to take previous knowledge from other classes that we have taken, especially circuits to think of ways to add functionalities to our design. This lab was very successful, and it not only reinforced the knowledge that we learned in this class, but we also combined knowledge from outside sources.

7. FIGURES








E
EEC172_SNS2 Mar 4
to me
Warning: It's too hot around the CC3200!