Temperature, Humidity, and Light

Difficulty: 2/5

Introduction

Temperature and humidity measurements are used widely in industry and manufacturing, and even at home. For example, the temperature and humidity conditions in a greenhouse could be critical for plant development. Being able to ***remotely*** monitor and record temperature and humidity is important because it can be very costly and inconvenient for a person to travel to sites to record these manually. In this tutorial, we will be using the Pete-Kit to create a remote temperature and humidity monitoring system.

This tutorial will also introduce you to analog sensors by using an ADC and an analog light sensor. Analog sensors are converted to digital signals that can be used by a computer (like the Raspberry Pi) by using an Analog to Digital Converter (ADC).

Part One: The Circuit

Parts List

For this project, you will need:

- Raspberry Pi
- Temperature/Humidity sensor module (AHT20)
- Light Sensor
- ADC Pi Hat
- Grove 4-pin female jumper cable

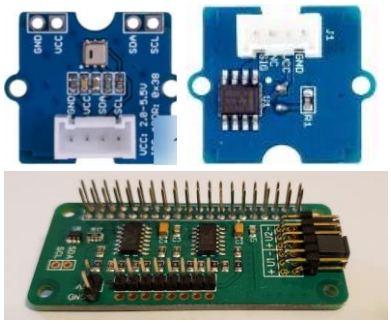

Figure 1: Temperature and humidity sensor (top left), light sensor (top right) and ADC Pi Hat (bottom).

Assembling the Circuit

In this tutorial, there are two devices to be connected to the Raspberry Pi. The temperature / humidity module is powered by the Raspberry Pi and also sends its data to the RPi. The Light Sensor is also powered by the Pi indirectly through the ADC Pi Hat. The ADC Pi Hat is mounted directly on top of the RPi, with pass through pins extending the existing pins on the RPi. The RPi is connected to the temperature / humidity module via the Grove 4-pin jumper cable. The ADC Pi Hat connects reads a signal from the light sensor and sends it to the RPi.

The temperature / humidity module has two signal connections (labeled SDA and SCL). These pins are used to output the measured temperature and humidity data from the sensor. The light sensor has one signal pin (labeled SIG). The “NC” connection on the light sensor means there is “no connection.” In other words, it doesn’t do anything.

The other two pins (labeled VCC and GND) on both sensors are the voltage wires used to power the sensor module.

- Plug a Grove cable into the temperature / humidity sensor.
- Plug a Grove cable into the light sensor.

Extra Info: It's not important right now to carefully understand the purpose of the SDA and SCL pins. However, they are the data and clock signals of a communication method called I^2C.

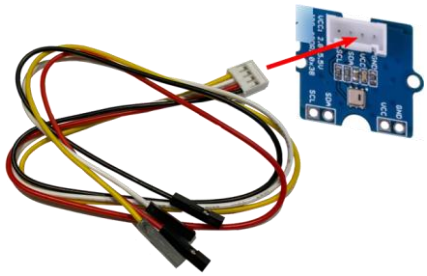

Figure 2: Connecting the grove cable to the temperature and humidity sensor.

When the Grove jumper cable is attached to the temperature / humidity sensor, the yellow and white wires connect to SCL and SDA pins, and thus carry the temperature and humidity data from the sensor to the Raspberry Pi. When the Grove jumper cable is attached to the light sensor, the yellow wire connects to the signal pin, carrying the signal from the light sensor to the Raspberry Pi. The red and black wires connect power to the temperature / humidity sensor and the light sensor.

- First, install the ADC Pi Hat on top of the RPi. It connects by the 40 pin connector. Start by opening the top of the Pi's enclosure. You can just pop it off by gently pressing in the side of the Pi enclosure opposite the USB ports. Just line up the 40 pins on the RPi with the 40 holes on the Hat and press the Hat onto the RPi until all the pins are fully seated (Figure 3 shows the Pi Hat installed on the RPi).
- Next, you need to connect the other ends of the Grove jumper cables to the Raspberry Pi. Figure 3 shows all the all the Grove jumper cables connected on the ADC Pi Hat.

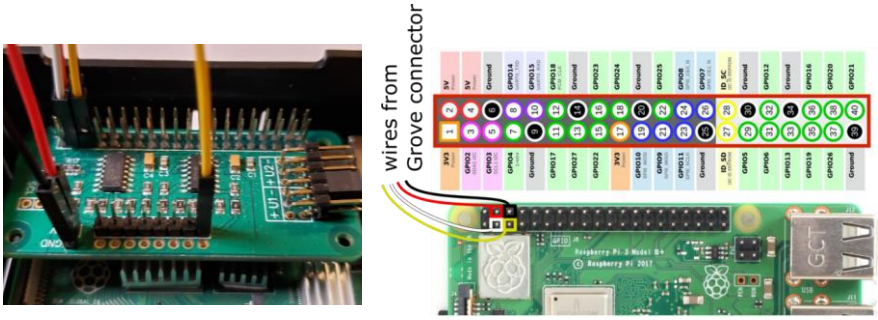

Figure 3: (left) picture of Grove jumper wires connected to ADC on the Raspberry Pi. (right) a "pinout diagram" is another way to view how the wires connect to the Raspberry Pi pinouts.

- Figure 3 (right side) shows which of the Raspberry Pi's pins the temperature/humidity sensor will connect to in the diagram on the right. Notice in Figure 3 (left side) that the

four wires from the temperature/humidity sensor are connected to the Pi Hat. These connections "pass through" the Pi Hat pins and also connect directly to the RPi.

- For the temperature/humidity sensor, notice how the red wire is connected to pin 4, the black wire is connected to pin 6, the yellow wire is connected to pin 5, and the white wire is connected to pin 3.
    - Pin 4 is a +5 V_{DC} pin for power
    - Pin 6 is the ground pin, also for power
    - Pins 3 and 5 are used for communication of data between the sensor and RPi.
- There is also a second set of connections for the light sensor on the ADC Pi Hat, which can be seen near the bottom of the left image in Figure 3. Connect the light sensor's red, black, and yellow wires as shown. Leave the white wire disconnected.
- Figure 4 shows the completed wiring with the ADC Pi Hat and both sensors attached to the RPi.

Figure 4: Completed hardware setup for temperature / humidity and light sensor project. Note you still need to hook up your Raspberry Pi to a monitor, keyboard, mouse, and power supply.

Part Two: Installing The Code

This section will describe how to get the software side of this project working. It is important to note that the code provided is not the only possible solution. You can find the sample code in the GitHub repository (located here: https://github.com/glcaptain00/Pete-Kit/) in the "Temperature" Folder. The name of the sample code file is "main.py" and contains the code discussed below. If you're interested in what the code does, in greater detail, see the Part Three.

Commented [JO1]: This folder doesn't exist on git hub yet.

Working sample code has been provided and is outlined in the following steps so that it is easier to follow.

Install Libraries

For your code to work, you need to install the necessary libraries. It is assumed your RPi is already connected to the Internet. If not, see our tutorial on how to do this (HERE). To install libraries, start by opening a terminal. If necessary, see the note below on how to open a terminal. In the terminal, run the following command by typing the following three commands given below. The ↵ symbol means you type the enter key.

```
sudo apt install git -y ↵

sudo pip3 install
git+https://github.com/abelectronicsuk/ABElectronics_Python_Libr
aries.git ↵

pip install smbus2 ↵
```

These commands will install the git system package, then the ADCPi and SMBus 2 libraries through the Python package installer. We are actually downloading the ADCPi library from its GitHub library, which is why we needed to install git! You may be asking "What is pip?" Go the Glossary to find out!

The libraries contain pre-written code that allows the RPi to use the ADC Pi Hat and I^2C communication bus correctly. It is very common to use pre-developed code as building blocks for projects. There is no need to re-invent the wheel!

Get and Store Code File

Copy main.py onto the RPi by opening the code on the GitHub repository. Then click "Raw" at the top right of the code. This will take you to a new page where you will only see the code as black text on a white background. Right click on the background, the click "Save as." A window will pop up, allowing you to choose a location to save the file. You can also view our video of this process by following this link: https://www.youtube.com/watch?v=VURIS1IOZEA

Note: Opening a Terminal

A terminal is a window that you can use to type commands that the Raspberry Pi will run for you. The easiest way to open a terminal is by first finding the file you want to run commands on and then selecting open terminal. To do this, open the file explorer and navigate to the file you want to work with. Once here, right click on the background of this window, and a pop up will appear with a list of selections. One of these selections will include "Open in terminal". Choose this, and you will see a terminal open. If you now type "ls" and press enter, you will see a list of files that the terminal has direct access to. This should include the file that you were looking for.

Commented [JO2]: I think this needs more explanation. 1. they need to get the Rpi going and then open a command window to type these commands. 2. they need internet access (I think). And if so, they need to know how to connect the RPi to the network. So minimally this tutorial should reference another for setting up the network and such. We need this to be as self-contained as possible, and where it cannot be, we need to provide instructions on where to find the solution.

Commented [CL3]: This is talked about in the section "Installing Libraries" first. I'd have suggested a move, but I'm not sure how to suggest a move.

Commented [JO4R3]: see bleow

Part Three: What The Code Does

THIS SECTION IS OPTIONAL

SAMPLE CODE IS PROVIDED AND WORKS WITHOUT EDITING. THIS SECTION IS JUST AN EXPLANATION OF THE SAMPLE CODE.

Looking at the Raspberry Pi Code Overview

First, let's look at the bigger picture. What will the code do? Making software is like making a building. You first need a plan or blueprints before you can start. Planning is an important part of making code do useful things.

The sample code you've opened (main.py) is already complete. **You don't need to change it!** It is designed to do the following tasks. If you do not know what some of the words mean, check the Glossary!

The steps the code will follow in this tutorial are as follows:

Task 1: Perform initial setup of devices attached to the RPi (not the RPi itself)
Task 2: Take a measurement
Task 3: Wait for a period

Looking at the code in detail

With this plan in place, we are now able to begin programming. We will begin with setting up devices. To do this, we need to perform the following steps.

1. Import 'smbus' library
2. Setup variables for easy reference to bus channel and device addresses
3. Initialize I^2C bus channel
4. Initialize temperature sensor.

In the next step, we will take measurements. The temperature sensor and ADC require specific commands to take a measurement.

1. Send start measurement command to temperature sensor
2. Send start measurement command to ADC
3. Read measurement from temperature sensor
4. Read measurement from ADC

Task 3, waiting for a period, is performed to minimize power usage. Instead of constantly drawing power to read measurements, we can afford to wait, as the temperature and humidity will not change significantly in 1/1,000 of a second.

Finally, we want the code to run forever (or until we stop it), so we are going to use a while loop and set the loop condition to true. This loop will contain Tasks 2 and 3.

Part Four: Running the Code and Viewing the Results

Now that everything has been setup, we can run the code and see it in action. For a detailed explanation of how to run the code, see below. If your code has errors and fails to run, you will get notified. Do not let this alarm you! The errors describe what happened and where, so take the time to read the message. It may help to view our video on this part too! You can see this video by following this link in your web browser: https://www.youtube.com/watch?v=VURIS1IOZEA.

How to run the code

If you have not run any code before, this should help you get started. This process applies to running any code file you want to. First things first, you'll need to make sure you know where your code file is! If you can't find it, you can't run it! It should still be where you saved it in Part Two of this tutorial.

Now, you'll need to open a terminal. One way to open a terminal is from the file browser. In the browser, navigate to the **folder** containing the file. Once you've done that, you can right-click on the window (not on the file!) and in the popup you should see "Open in Terminal" or "Open Terminal here." Click this, and a terminal will open in the proper folder.

Now, you need to type *sudo python3* followed by the name of your file. For example, if your file is named "main.py" (the original name of it), then you would type the command:

```
sudo python3 main.py
```

We need to use the sudo command here so that the code has permission to access the GPIO pins. Only programs with administrative access can do this (this is a security thing) and sudo gives our code administrator rights.

Check that everything is working

If your code does not have errors, it is now time to verify the proper performance of your program. This project does not require any interaction. The sensors on it are always able to take a value, and the Raspberry Pi will handle those measurements for you. Instead, all you need to do is run the code! You will see readings of the temperature, humidity, and light levels being output to the console! These values should update once every second.

Seeing the values

The measurements from the sensors are read by the code and output to the console. Try covering the light sensor to watch its value change! You can also hold the temperature / humidity sensor in your hands and breathe into your hands as if you were trying to warm them on a cold day. You should see the temperature and humidity values rise!