Lab 06: GrovePi Sensors

Code Submission: 11th & 13th February
Demo Checkoff: 11th & 13th February

New Lab Announcements -

- **Lab to be done in pairs(mandatory)**
- **Collect your GrovePi as soon as possible (VHE205 during lab sessions)**

Files will be submitted on Vocareum. Files to submit:

- **grovepi_sensors.py**
- **lab06_writeup.md (or .txt)**

1. Introduction

The purpose of this lab is to practice using your Raspberry Pi and interface it with the GrovePi+ starter kit, which includes an RPi shield that can connect various Grove sensors using standardized 4-pin connectors to your RPi. You will use the Python libraries provided by the makers of the GrovePi to interface with the sensors.

You will use the following components in your GrovePi Kit. Please let a TA know if you are missing any of these components.

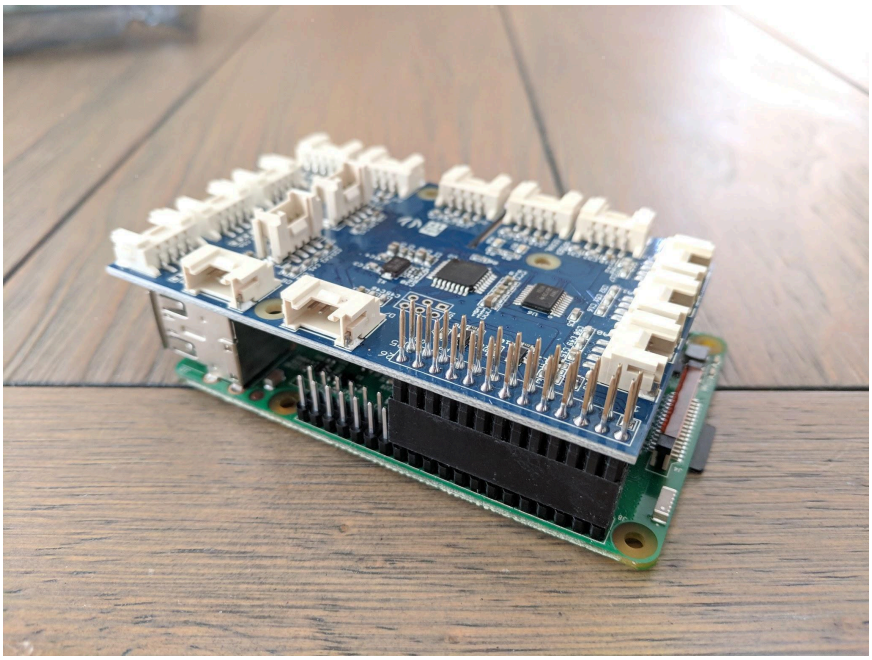
- GrovePi Board
- Grove – Sound Sensor
- Grove – Light Sensor
- Grove – Ultrasonic Sensor
- Grove – Rotary Angle Sensor
- Grove – LCD RGB Backlight
- Sensor Cables

It should also contain these components, but you will not use them.

- Grove – Temperature and Humidity
- Grove – Relay
- Grove – Button
- Grove – Buzzer
- Grove – Red LED
- Grove – Blue LED
- Grove – Green LED

2. Updating & Testing the GrovePi Shield's Firmware

First, please attach the GrovePi shield to your Raspberry Pi (while powered off). This step is important because you are not only installing software on the RPi, but you are also flashing the microcontroller on the GrovePi itself.



The environment inside your RPi needs to be set up for the GrovePi board, and the firmware in the Atmega328P processor used on the GrovePi board itself needs to be updated to support the ultrasonic sensor. Because GrovePi has created very specific scripts with a lot of hardcoded paths, we will have to run the installation procedure using their method.

Then, you need to execute their installation shell script written in the "bash" shell-scripting language. This is why there is a pipe "|" followed by "bash" towards the end of the command. What this essentially does is pipes the output of the `curl` (which is the downloading of the bash script) and executes it via bash. **Pro tip**: to paste inside Ubuntu's Terminal program, use ctrl+shift+v.

Step 1:

Ssh into your Rpi

Step 2 in your terminal:

sudo apt-get install curl

Step3 in your terminal:

curl -KL
https://raw.githubusercontent.com/Tamoghna-Sarkar/FixGrove/main/grove-setup.sh* | *bash

Step4:

cd /Dexter/GrovePi/Firmware

Step5:

sudo raspi-config nonint do_spi 1
sudo raspi-config nonint do_i2c 1
./firmware_update.sh

Step6:

TEST YOUR GROVEPI w/ your LED

Attach the LED to Digital Pin 4 on the GrovePi (you can change the pin number according to where you attach yours, I used 4)

cd Dexter/GrovePi/Software/Python

python grove_led_blink.py

(make sure the +ive and -ive terminals are inserted accordingly)

And now you should see your LED blinking.

NOW, GET STARTED WITH YOUR LAB6!

[OPTIONAL] To check the firmware, you can run the following commands

```
cd ~/Dexter/GrovePi/Software/Python
python grove_firmware_version_check.py
```

You should see output like the following:

```
GrovePi has firmware version: 1.4.0
```

3. Python Programming Task

There are two typical strategies you can use to approach coding on the Raspberry Pi:
The first is to make changes to the code in your VM, `git push` the commits to your GitHub repository from your VM, and, in your rpi `git pull` the changes to run.
The second is to directly modify the files on your rpi via a command-line text editor like `nano` (or install `micro`,`emacs`, `vim`, etc.) and `git push` any committed changes from the raspberry pi to Github.com (and then pull them back to your VM).
This is up to you. The main task associated with this lab will be run on the Rpi along with the GrovePi and the sensors.

In this lab, you will explore and figure out how to use the Python libraries provided by Dexter Industries to interface with the various sensors and actuators in the GrovePi+ starter kit. In particular, you will use these devices:

- Grove Ultrasonic Ranger (plug into any digital port)
- Grove Rotary Angle Sensor (aka potentiometer, plug into any analog port)
- Grove LCD RGB Backlight (can be plugged into any i2c port)

Your task is to write a program that

1. Lets you set a threshold distance by turning the rotary angle sensor
2. Measures the distance to an object using the ultrasonic ranger
3. Determines whether the object is within the threshold distance

Your program will also print out to the LCD screen three pieces of information:

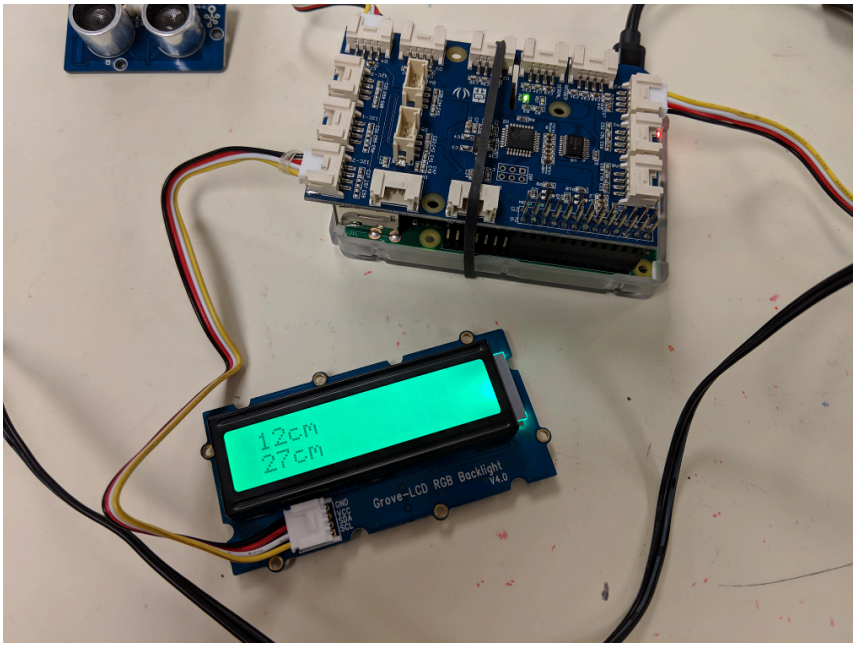
1. Current threshold value (top line)
2. "**`OBJ PRES`**" (top line) if the ultrasonic ranger output falls below the threshold value
3. Current ultrasonic ranger measurement (bottom line)

The integer value returned by the Python function calls provided by Dexter Industries represents the distance in centimeters. Because these ultrasonic rangers are cheap and not very accurate, you may see a noisy output.
The threshold range should be mapped directly to the full range of the rotary angle sensor output values, which is [0, 1023]. While we tested the ultrasonic ranger, we found that it exhibited an output in the estimated range of [0, 517] with a max depending on the size and dynamics of the room.

Formatting of the LCD screen prints must adhere to the following structure:

- The top line should display the current threshold value followed by a space.
- If an object is present (that is, the raw ultrasonic sensor output falls below the threshold), then the string "**`OBJ PRES`**" should be printed to the right of the current threshold value on the top line, separated by a space.
- The bottom line should print out the current raw ultrasonic ranger output.
- Lastly, you should use the function call that does not refresh the LCD screen when setting the text.

No object detected closer than the threshold:



Object in range (changing backlight color is not required):

We have provided a starter file in the drive named "grovepi_sensors.py"

`grovepi_sensors.py`

Please code inside this single file for Lab 06. Place this file inside your Rpi and complete the script.

You are responsible for sifting through their example code folder to find relevant examples and studying them. Look in **~/Dexter/GrovePi/Software/Python** to find these examples. **As a tip, try to modify and run the example code**
(*Note: In an internship or other real-world experience, this is what would be expected of you. You will be given some sample code and asked to read the documentation and then left to your own devices. Put the time and effort in on this exercise and you'll feel more comfortable in the future.*).
There are some resources online you can find via Google and some resources in the pamphlet inside the GrovePi boxes. It is well known that software documentation can often be confusing and incorrect (Dexter Industries is notorious for this), and this is a typical problem in the world of engineering that we want students to gain experience navigating.

4. Reflection Questions

Answer all of these questions in a text file named “lab06_writeup.md”. You will submit this text file as a part of Lab 06.

4.1. Suppose you just cloned a repository that included one python file, my_first_file.py, and you now want to add a second file to your repository named my_second_file.py which contains the following code **and push it to Github.com**.

Code -

print(“Hello World”)

Complete the sequence of linux shell commands:

```
git clone git@github.com:my-name/my-imaginary-repo.git
##complete the sequence
```

(Note: create the file using the `touch` command)

4.2. Describe the workflow you adopted for this lab (i.e. did you develop on your VM and push/pull to get code to your RPi, did you edit files directly on your RPi, etc.). Are there ways you might be more efficient in the next lab (i.e. learning a text-based editor so you can edit natively on the RPi, understanding Git commands better, etc.)?

4.3. In the starter code, we added a 200 ms sleep. Suppose you needed to poll the ultrasonic ranger as fast as possible, so you removed the sleep function. Now, your code has just the function ultrasonicRead() inside a while loop. However, even though there are no other functions in the while loop, you notice there is a constant delay between each reading. Dig through the python library to find out why there is a constant delay. What is the delay amount? In addition, what communication protocol does the Raspberry Pi use to communicate with the Atmega328P on the GrovePi when it tries to read the ultrasonic ranger output using the `grovepi` python library?

5. Demo and Code Grading Rubric

All files are to be submitted via Vocareum.

Points	Description
DEMO	
2	The analog rotary angle sensor controls the threshold with the range [0,1023], and the current threshold is displayed properly on the LCD.
2	The current ultrasonic ranger raw output is displayed on the LCD in centimeters.
4	“OBJ PRES” is printed when the ultrasonic ranger outputs a value lower than the current threshold.
2	The LCD does not refresh every time new text is displayed. (The LCD will appear to be blinking if the LCD is refreshing.)
CODE	**Each team member should submit their respective grove_sensors.py**
2	Team Member names at the top of the script to correctly assign points to both(**both should submit**)
2	The formatting of data displayed on the LCD follows the specifications.
2	Code correctness (no python syntax errors, sufficiently bug-free for the assignment, etc.)
Reflection	**Submit a text file named lab06_writeup.md with your answers to the questions. Demo during checkoff by next Lab Session.**
2	Q4.1 (see section 4 - Reflection Questions)
1	Q4.2
1	Q4.3
	Total Possible: 20

Other Comments:

What is RGB?
https://www.w3schools.com/colors/colors_rgb.asp

Here are some common errors during the firmware flashing part:

1. Avrdude complains about loading shared libraries.

    avrdude: error while loading shared libraries: libusb-0.1.so.4: cannot open shared object file: No such file or directory
    avrdude: error while loading shared libraries: libusb-0.1.so.4: cannot open shared object file: No such file or directory
    avrdude: error while loading shared libraries: libusb-0.1.so.4: cannot open shared object file: No such file or directory
    avrdude: error while loading shared libraries: libusb-0.1.so.4: cannot open shared object file: No such file or directory

    To fix this, try sudo apt-get --fix-broken install.