Cohort Session 3, Week 12
GUI with Kivy

Objectives

1. Learn the fundamentals of GUI programming through the use of Kivy.
2. Make use of Firebase in a GUI to connect devices.

Please work on this lab in a group of **two or three**. Be sure to email your partner all the modified code, printouts and data. You may have to use them during your exams.

1 Equipment & Software

Download the software for this project from **1D Project website**.

Each group should have:

1. A Raspberry Pi.
2. A cobbler.
3. An LCD touch screen.
4. A wireless keyboard.
5. A wireless mouse.
6. One red LED.
7. One yellow LED.
8. Two resistors (330Ω).
9. A few jumper wires.
10. A Firebase account with a database and a secret token.
11. Installed Kivy according to the instructions on **Kivy Installation Guide**.

For MAC users, you will need to download the `firebase.py` from **Courseware website**, put it in your handout folder and import firebase as per normal.

2 Controlling LEDs with GUI

Task
Create a Kivy application to control two LEDs on the Raspberry Pi.

Figure 1: Kivy application.

Write a Python program using the Kivy API to create a GUI application as shown in figure 1. This application should be running on your laptop. The application is used to control two LEDs that are connected to the Raspberry Pi. Write another Python program running on the Raspberry Pi to control the LEDs.

On the Raspberry Pi, you should wire up one yellow LED and one red LED along with current-limiting resistors. Whether or not the LEDs light up will depend on the value of the data on the Firebase database.

The GUI should contain:

1. two Labels that specifies the yellow and red LED.
2. two ToggleButtons to toggle the state of each LED on the Raspberry Pi. The ToggleButtons can either be `on` or `off`. For example, when the ToggleButton beside the Label that reads yellow LED is `on`, then the physical yellow LED should be switched on. (You can refer to **ToggleButton Documentation** for ToggleButton usage. Note that you can also use `on_press` as in the Button class.)

The GUI application on your laptop should write the state of the LEDs to the Firebase database so that they can be read by the Python program on the Raspberry Pi. Furthermore, the application should also check the database on start up so that the state of the toggle buttons corresponds to the data on Firebase. Meanwhile, as long as the Python program on the Raspberry Pi is running, the LEDs should always correspond to the value of the data on Firebase except for the initial state where the data has yet to be written to the Firebase by the GUI application. During the initial state, the LEDs will remain off.

Here are some examples of the expected behaviour of the two programs:

The very first time the GUI application and the Python program on the Raspberry Pi is started, the LEDs on the Raspberry Pi should remain off. The application should appear as in figure 1.

When a user presses the toggle button corresponding to the yellow LED, the yellow

LED should switch on. The red LED remains off. The application should appear as in figure 2.

Figure 2: Kivy application.

When the user closes the GUI application at this point, the state of the physical LEDs should not change. That is, the yellow LED remains on and the red LED remains off.

When the user restarts the GUI application at this point, it should appear as in figure 2 instead of figure 1.

When the user presses the toggle button corresponding to the yellow LED, the yellow LED should switch off. The red LED remains off. The application should appear as in figure 1.

You may refer to the following figure to wire up the the LEDs and switch.

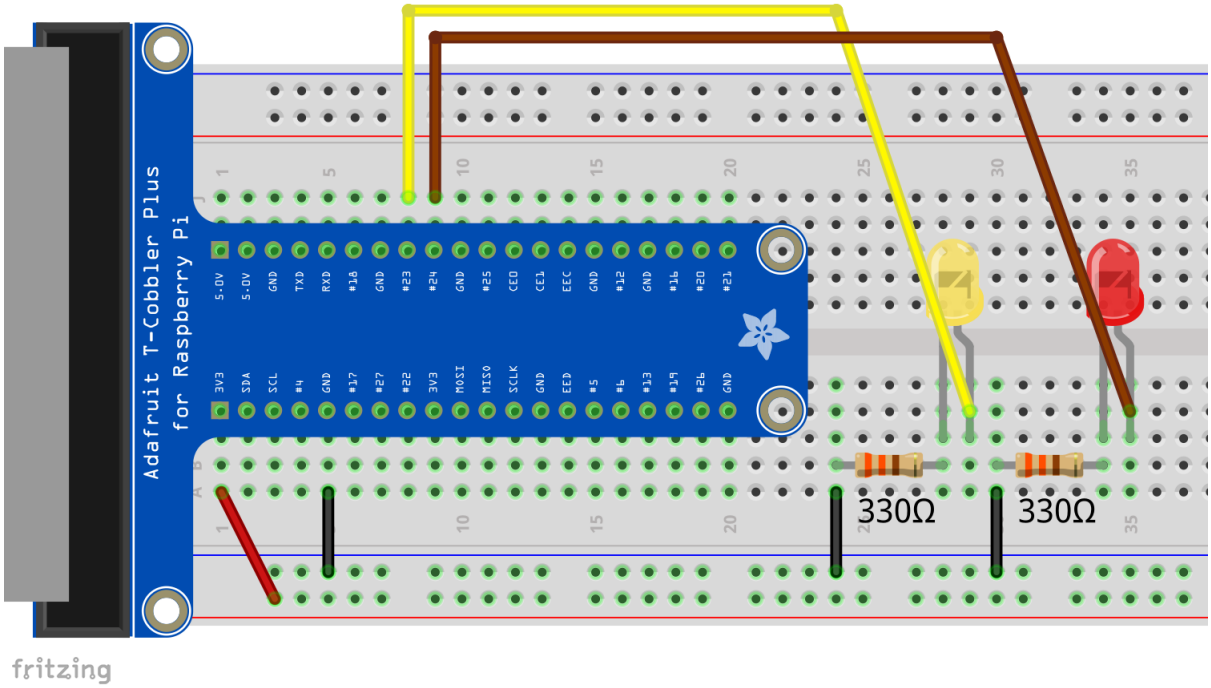

Figure 3: Note that the longer leg of the LED is the positive end. Hence it should be connected to the GPIO pin.

WARNING!
The state of the GPIOs will be retained after the program terminates. That is, if any GPIO output was HIGH at the moment the program terminates, it would remain as HIGH even after the program terminates.

Consequently, it is possible to accidentally damage the Raspberry Pi by connecting GPIOs that was HIGH directly to ground. In order to 'reset' the GPIOs, you can use gpiocleanup.py (found at **Courseware website**) and run it in your Raspberry Pi. The program sets every GPIO as input. As such, they would not get damaged even if they are connected directly to either HIGH or LOW.

Checkoff 1
Explain and demonstrate the working program to an instructor. Your programs should:

1. use Kivy to create a GUI application
2. use Firebase to read and write the state of the LEDs.