Camera

Difficulty: 2/5

Introduction

Cameras are used in many situations, ranging from capturing memories to security. In this tutorial, we will be using the Pete-Kit to create a basic camera, with a button used to capture the picture.

Part One: The Circuit

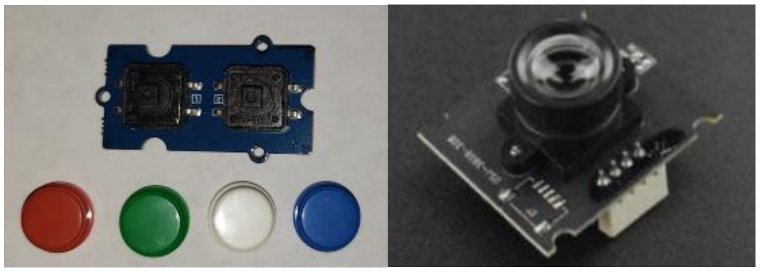

Figure 1: Dual button module (left) and USB camera (right).

Parts List

For this project, you will need:

- Raspberry Pi
- USB Camera Module
- Dual Button Module
- Grove 4-pin female jumper cable

Assembling the Circuit

In this tutorial, there are two devices to be connected to the Raspberry Pi.

The USB Camera module sends pictures to the Raspberry Pi. They are connected by the camera's USB cable.

- Plug the camera into **any** one of the Raspberry Pi USB ports.
- See Figure 2. Your cable will be longer!

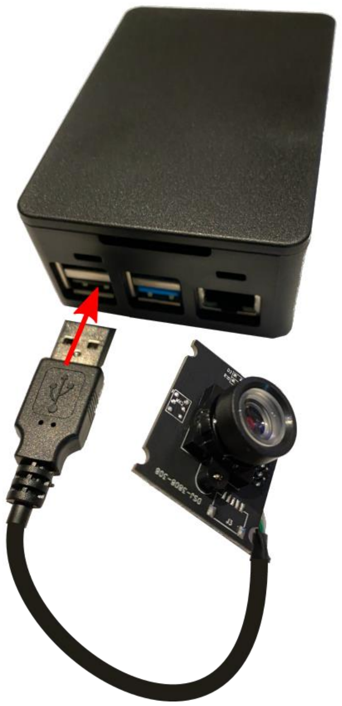

Figure 2: Connecting the USB camera to the Raspberry Pi.

The Dual Button Module has two signal connections (labeled SIG1 and SIG2). When the Grove jumper cable is attached, the yellow and white wires connect to SIG1 and SIG2 and allow the buttons to communicate with the Raspberry Pi. The red and black wires connect power to the Dual Button Module.

- Connect the Grove cable to the dual button module using the white connector. It should slide in and clip easily.
- See Figure 3 for a closer picture of the various parts.

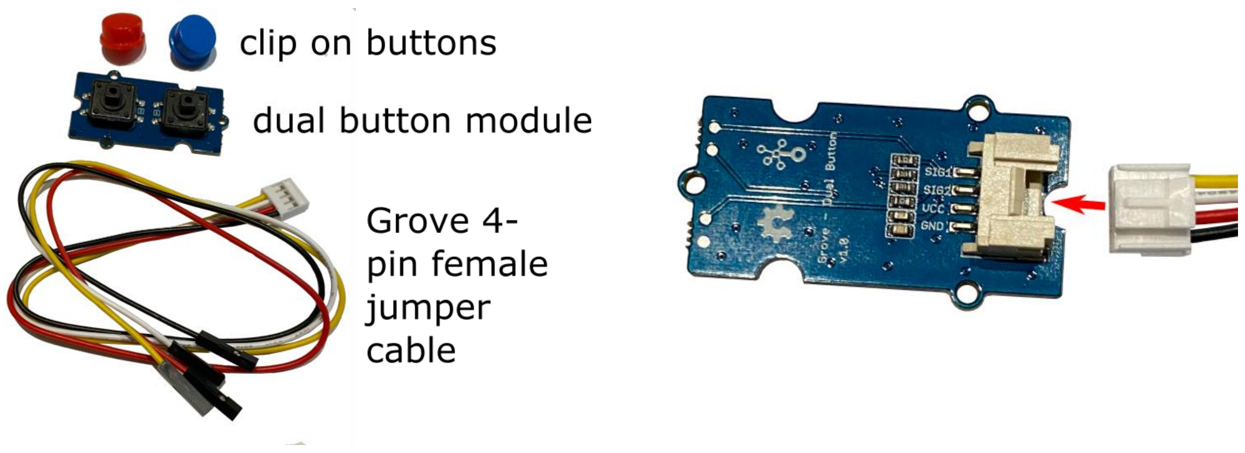

Figure 3: (left) Dual button module, clip on buttons, and Grove 4-pin jumper cable. (right) connecting the Grove jumper wire to the dual button module.

- Next, you need to connect the other end of the Grove jumper cable to the Raspberry Pi. 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. See Figure 4.

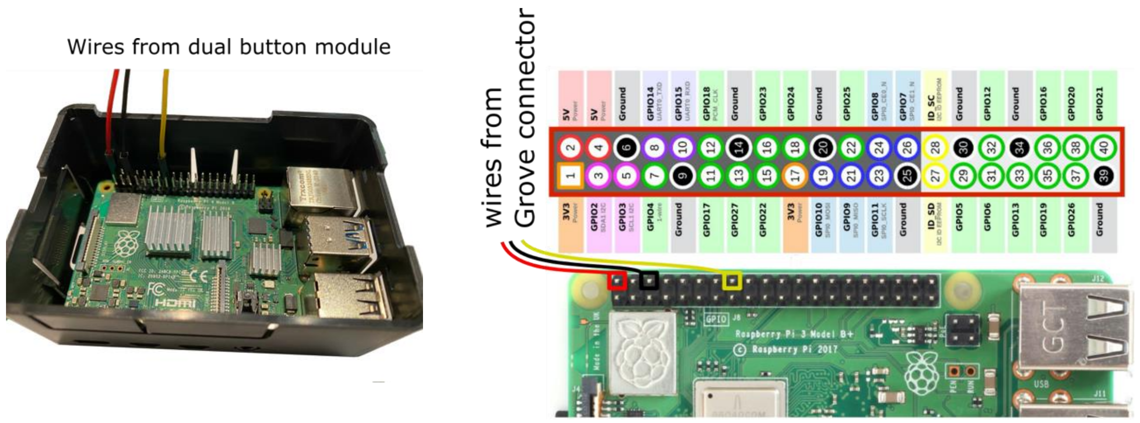

Figure 4: (left) picture of Grover jumper wires connected to Raspberry Pi. (right) a "pinout diagram" is another way to view how the wires connect to the Raspberry Pi pinouts.

- Notice how the red wire is connected to pin 2, the black wire is connected to pin 6, and the yellow wire is connected to pin 16.
- Pin 2 is a +5 V_{DC} pin for power
- Pin 6 is the ground pin, also for power
- Pin 16 is a GPIO pin, specifically GPIO number 23.
- The white wire is not connected anywhere, which means one of your buttons on the dual button module will not do anything.

Figure 5 shows an image of all the connections. GPIO pins allow signals to be passed back and forth between the Raspberry Pi and a connected device. In this case, the Raspberry Pi will be receiving a signal from the button.

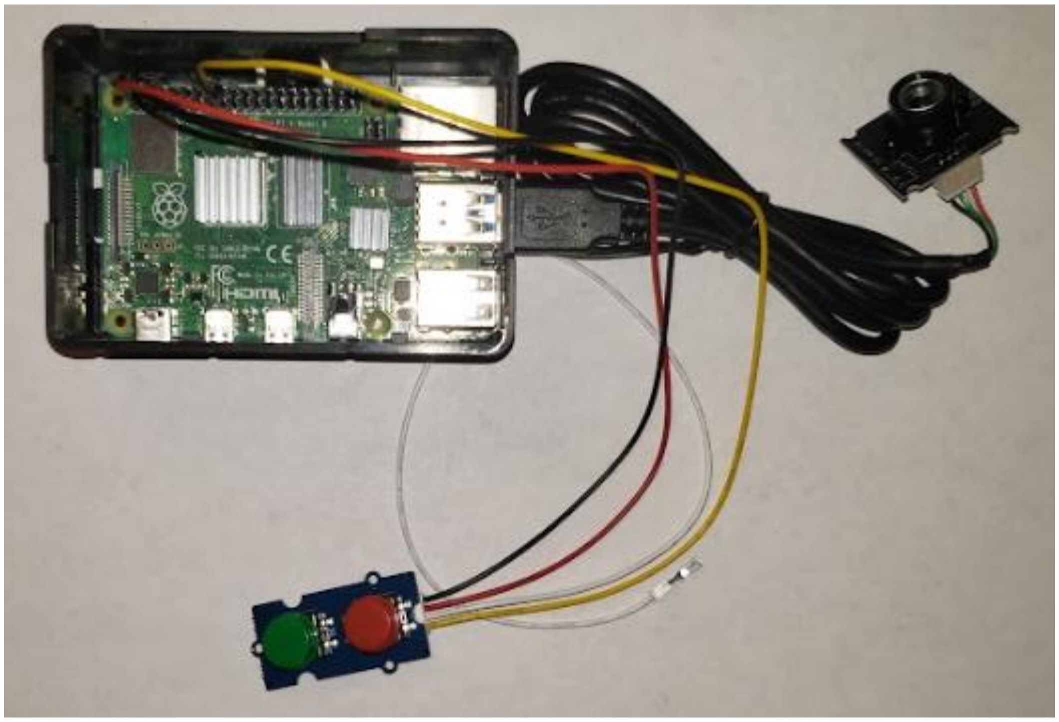

Figure 5: Completed hardware setup for camera project. Note you still need to hook up your Raspberry Pi to a monitor, keyboard, mouse, and power supply.

Part Two: The Code

The section below will describe the thought process used in designing the sample code for this project. You do not need to read that section unless you would like to. 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/blob/main/Camera/main.py) in the “Camera” Folder. The name of the sample code file is “main.py” and contains the code discussed below.

Install Libraries

For your code to work, you need to install the necessary libraries. To do this, start by opening a terminal. If necessary, see the note below on how to open a terminal. In the terminal, run the following commands by typing what is given, one line at a time, then pressing enter after each lin.:

```
pip install opencv-python
sudo apt-get install python3-rpi.gpio -y
```

These commands will install the OpenCV library through the Python package installer and the RPi’s GPIO library through the system package manager. You may be asking “What is sudo?” or “What is pip?” Go the Glossary to find out!

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 (NEED VDIEO HERE).

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.

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:	Wait for a button press
Task 3:	Take a picture, then save the picture to the current directory

Looking at the code in extreme detail

With this plan in place, we are now able to begin programming. The first task is to setup devices. This will include importing the appropriate libraries. This step begins on line 2 of the sample code.

1. Code imports cv2 (for the camera) and datetime (For naming the pictures) libraries
2. Code selects a GPIO Pin for the button input, and stores it in a variable
3. Code set up the camera in video capture mode

The next step is to wait for the button press. This can be done using loops. This step begins on line 14.

1. Code uses a while loop with the condition that checks for the button NOT being pressed
2. In the while loop, the code adds a "sleep" function to pause for a short time.

The next step is to take a picture. The library cv2 makes this easier. This step begins on line 54 of the sample code.

1. Code reads the current frame of video from the camera
2. Code creates a string representation of the current time to use as the filename, then stores this string in a variable. What is a "string"? See your Glossary.
3. Code uses the cv2 library to write the image to a file (save it).

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

Part Three: Result

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 scare 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 (**NEED VERY SHORT VIDEO HERE**).

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. Another 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. 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. Press whichever button you connected to pin 16, then look for image files in the same directory as your python code. If you see the images, open them, and make sure they are properly saved.

Find and Display the Stored Pictures

The pictures taken by the camera will be stored on your Raspberry Pi in some folder. You can view the pictures by navigating to that folder and double clicking the picture files. The picture files will be named by the date and time they were recorded, but you can rename them if you like.