Camera

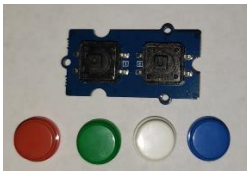
Difficulty: 2/5

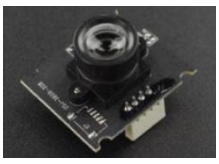
Introduction

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.

Parts List

For this project, you will need:





- Raspberry Pi
- USB Camera Module
- Dual Button Module

Code Requirements

This project uses the following libraries:

- OpenCV – Open source computer vision library
- Datetime – Library for working with dates and times
- RPi.GPIO – Library for utilizing the Raspberry Pi's GPIO pins

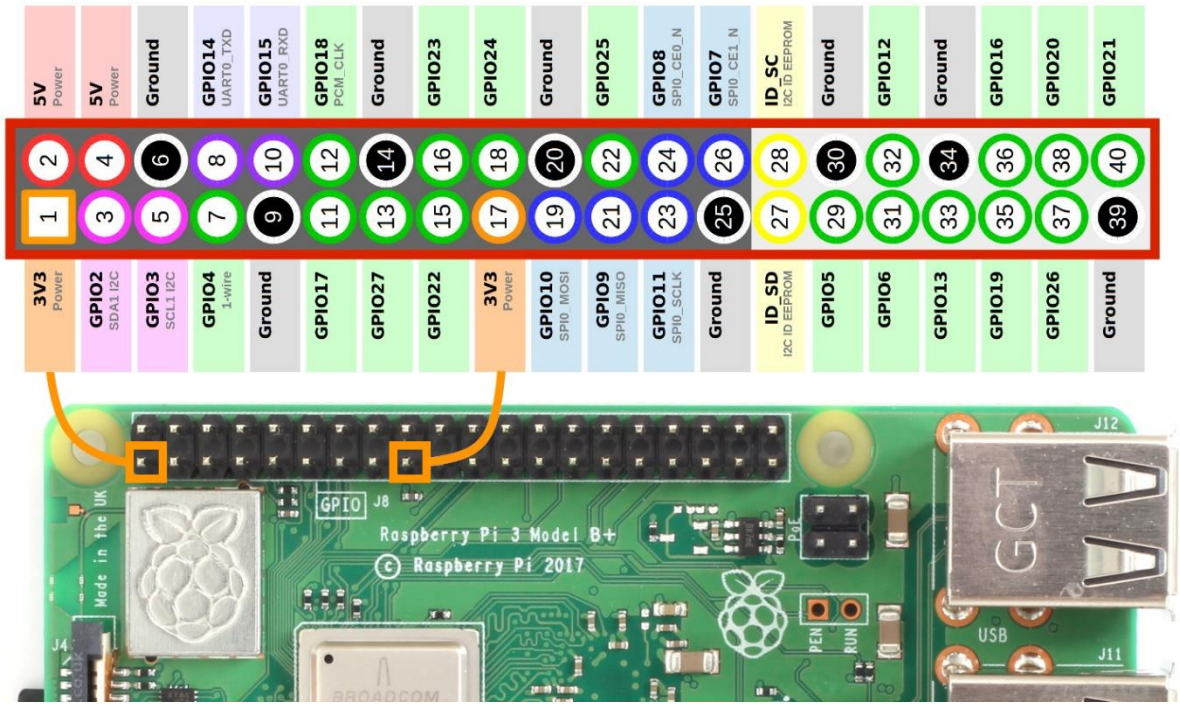
The Datetime library is already installed. To install the other libraries, open a terminal on the Raspberry Pi and run the following commands:

```
pip install RPi.GPIO
pip install opencv-python
```

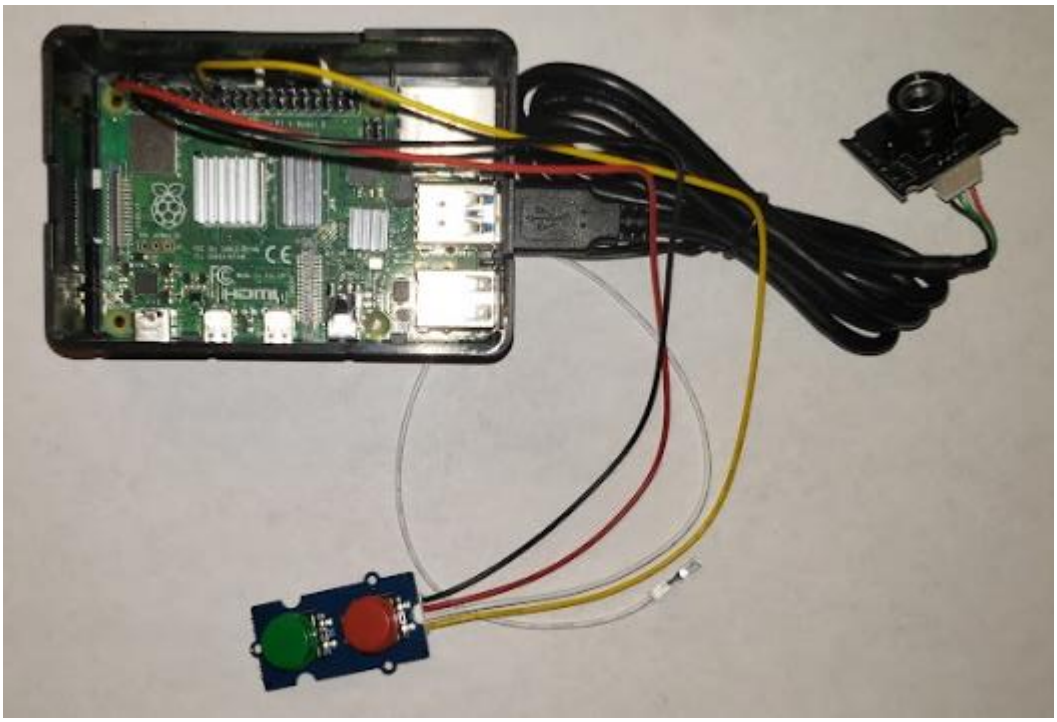
Part One: The Circuit

In this tutorial, there are two devices to be connected to the Raspberry Pi. The USB Camera module can plug into any of the four USB ports. For the Dual Button Module to work with the sample code, either of the signal pins (SIG1 or SIG2) need to be connected to pin 16.

The GND pin connects to a black wire and the VCC pin connects to a red wire. These are typical colors to represent these connections, which are used for power. The GND pin will always connect to the ground pin of the Raspberry Pi, and the VCC will connect to a 5v Power pin. The SIG1 and SIG2 pins are signal pins, and will tell the RPi when the button has been pressed. Below is a "pinout diagram." This means that the below image shows you which connection each pin makes.



Pay attention to the numbers inside of the red rectangle. These are the physical, or "board" pin numbers. These tutorials will refer to the board pin numbers. Below you will see an image of the connections. The red wire is connected to pin 2 (for a 5V connection), the black wire is connected to pin 6 (for a ground connection), and the yellow wire (SIG1) is connected to pin 16. Pin 16 is connected to GPIO23. 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.



Part Two: The Code

This section will describe the thought process behind the sample code provided with this document. 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:) in the “Camera” Folder. The name of the sample code file is “main.py” and contains the code discussed below.

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. Import cv2 (for the camera) and datetime (For naming the pictures) libraries
2. Select a GPIO Pin for the button input, and store it in a variable
3. Setup 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. Use a while loop with the condition that checks for the button NOT being pressed
2. In the while loop, add a “sleep” function to sleep 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. Read the current frame of video from the camera
2. Create a string representation of the current time to use as the filename. Store in a variable
3. Using the cv2 library, write the image to a file.

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. To run your code, open your command line to the location of the python file. Then type “python” followed by the name of your file. For instance, if your file was named “Camera.py” you would type “python Camera.py” followed by pressing the enter key. If your code has errors, 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.

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.