Hardware Coding

(Haroldas Tamosauskas)

**To run the Raspberry Pi Camera Module 3 NoIR:**

The hardware would require an update for the Raspberry Pi in order to make it work properly, best case scenario is to update the Pi fully using these commands:

|  |
| --- |
| **sudo apt update**  **sudo apt full-upgrade**  **sudo rpi-update**  **sudo reboot** |

Afterwards, it is best to disable the legacy camera inside the Interface Options, then afterwards rebooting it and using the camera to take a picture:

|  |
| --- |
| **sudo raspi-config**  **sudo reboot**  **libcamera-still -o image.jpg -t 5000** |

**(5000 represents it takes 5 seconds to take the picture)**

**To run the Mini USB Microphone:**

This hardware just requires using this code to record for 5 seconds (As long as you updated Raspberry Pi before):

|  |
| --- |
| **arecord -D plughw:1,0 -d 5 test\_audio.wav** |

* **arecord: a command line utility for linux to record audio**
* **-D: Specifies the device**
* **Plughw:1,0: the name of the device (Mini USB microphone)**
* **-d 5: will record for 5 seconds**
* **Test\_audio.wav: will output as “Test\_audio.wav”**

**To run the Mini USB Microphone to the mini–External USB Stereo speaker:**

Before writing the code for connecting the mini USB microphone to the mini-External USB stereo speaker, it is best to update the Raspberry Pi Zero to support “sounddevice”, Here is how to do it:

|  |
| --- |
| **python3 -m pip install --upgrade pip (To check if pip is up to date)**  **python3 -m pip install sounddevice** |

Create the python file (for this case, it will be referred to as “microphone\_to\_speaker\_code.py”):

|  |
| --- |
| **nano microphone\_to\_speaker\_code.py**  **vim microphone\_to\_speaker\_code.py** |

Write the following code:

|  |
| --- |
| **import sounddevice as sd**  **# Specifies the devices IDs**  **input\_device = 1  # Microphone**  **output\_device = 0  # Speaker**  **# Sampling rate for your device**  **SAMPLE\_RATE = 48000**  **CHANNELS = 1**  **# Size of each audio block**  **blocksize = 1024**  **# Function to process audio input and send it to the output in real-time**  **def audio\_callback(indata, outdata, frames, time, status):**  **if status:**  **print(status)**  **outdata[:] = indata  # Copy input data to output**  **print("Start speaking... Press Ctrl+C to stop.")**  **# Sets up the audio stream**  **try:**  **# Opens a stream to read from microphone and write to speaker**  **with sd.Stream(samplerate=SAMPLE\_RATE, channels=CHANNELS, dtype='int16') as stream:**  **while True:**  **# Reads and writes the audio in chunks**  **data, \_ = stream.read(SAMPLE\_RATE // 10)  # Reads 1/10th of a second of audio**  **stream.write(data)**  **except KeyboardInterrupt:**  **print("\nExiting...")**  **except Exception as e:**  **print(f"Error: {e}")** |
|  |

Be sure to press, esc, : and the wq to save the code.

To change the audio (Optional):

|  |
| --- |
| **(For this particular device, the control\_name is PCM)**  **amixer -c 0 set <control\_name> 50% -for 50% volume** |

Now run the code and it should be able to detect the audio from the microphone to the speaker (Press ctrl + c to exit)

|  |
| --- |
| **python3 microphone\_to\_speaker\_code.py** |

**To run the Raspberry Pi Camera Module 3 NoIR live on Raspberry Pi zero:**

**(code gotten from: https://www.youtube.com/watch?v=U7yVpYv3gxQ)**

In order to run the live feed of the Raspberry Pi Camera module 3 on the Raspberry Pi zero, this is the code to use:

|  |
| --- |
| **libcamera-hello**  **libcamera-hello -t 0 (for a constant live feed)** |

In case that it does not detect the camera, it is best to enable legacy camera in:

|  |
| --- |
| **sudo raspi-config** |

Then later go to Interface Options > Legacy Camera, enable it and do:

|  |
| --- |
| **sudo reboot** |

If it still doesn't detect the camera, go to this config file:

|  |
| --- |
| **sudo nano /boot/config.txt** |

Be sure to add or uncomment these files:

* **camera\_auto\_detect=1**
* **dtoverlay=vc4-kms-v3d**

Also be sure to comment out these files:

* **#start\_x=1**
* **#gpu\_mem=128**

Ctrl+o and enter to save and exit, then ctrl+x to exit editor. Perform the sudo reboot and run libcamera-hello.

**To test the pir motion sensor:**

In order to test the pir motion sensor inside the Raspberry Pi zero, first thing you need to do is make a python file (for this case “motion\_test.py”):

|  |
| --- |
| **nano motion\_test.py** |

Then afterwards enter the file using vim motion\_test.py (or sudo vim motion\_test.py if there is not permission), then enter the following code:

|  |
| --- |
| **import RPi.GPIO as GPIO**  **import time**  **PIR\_PIN = 17**  **GPIO.setmode(GPIO.BCM)**  **GPIO.setup(PIR\_PIN, GPIO.IN)**  **print("Monitoring action")**  **try:**  **while True:**  **if GPIO.input(PIR\_PIN):**  **print("Motion!")**  **else:**  **print("No motion")**      **time.sleep(0.5)**  **except KeyboardInterrupt:**  **print("\nExiting program.")**  **finally:**  **GPIO.cleanup()** |

* **PIR\_PIN = 17 (The motion sensor is connected to GPIO 17 pin, could be changed to something else)**
* **if GPIO.input(PIR\_PIN): (If the motion is detected, print “Motion!”)**
* **time.sleep(0.5) (used to delay the detection)**
* **except KeyboardInterrupt: (If the user presses ctrl+c, it will exit and say “Exiting program.)**
* **GPIO.cleanup() (Resets all pins to inputs)**

Then run the code using the following command to run the code:

|  |
| --- |
| **sudo python3 motion\_test.py** |

If everything should work, it should read “Motion!” if there is motion, and “No motion” if there is no motion.

**To run the Raspberry Pi Camera Module 3 NoIR with the pir motion sensor:**

**(code gotten from: https://forums.raspberrypi.com/viewtopic.php?t=368757)**

In order to make it so that the motion sensor would activate the camera, record for 5 seconds and then save as a mp4, it is required to create a file, (in this case “motion\_picamera2.py”):

|  |
| --- |
| **nano motion\_picamera2.py** |

Inside motion\_picamera2.py, put in this code:

|  |
| --- |
| **from picamera2 import Picamera2**  **from picamera2.encoders import H264Encoder**  **from picamera2.outputs import FfmpegOutput**  **import RPi.GPIO as GPIO**  **import time**  **import cv2**  **from pubnub.pnconfiguration import PNConfiguration**  **from pubnub.pubnub import PubNub**  **# PubNub configuration**  **pnconfig = PNConfiguration()**  **pnconfig.subscribe\_key = "<subscribe-key"**  **pnconfig.publish\_key = "<publish-key"**  **pnconfig.uuid = "raspberry-pi-motion"  # Unique identifier for this client**  **pubnub = PubNub(pnconfig)**  **# Function to publish messages**  **def publish\_message(message):**  **pubnub.publish().channel("motion-detection").message(message).sync()**  **# PIR sensor setup**  **PIR\_PIN = 17  # GPIO pin**  **GPIO.setmode(GPIO.BCM)**  **GPIO.setup(PIR\_PIN, GPIO.IN)**  **# Initializes the Picamera2**  **picam2 = Picamera2()**  **video\_config = picam2.create\_video\_configuration(main={"size": (640, 480)})**  **picam2.configure(video\_config)**  **picam2.start()**  **print("Waiting for motion...")**  **try:**  **while True:**  **# Checks for motion from the PIR sensor**  **if GPIO.input(PIR\_PIN):**  **print("Motion detected!")**  **publish\_message({"event": "motion\_detected", "timestamp": time.strftime("%Y-%m-%d %H:%M:%S")})**  **# Creates a filename for the video**  **timestamp = time.strftime("%Y%m%d\_%H%M%S")**  **encoder = H264Encoder(10000000)**  **filename = FfmpegOutput(f"/home/haroldt2/motion\_{timestamp}.mp4")**  **# Starts recording the video**  **picam2.start\_recording(encoder, output=filename)**  **# Records video for 5 seconds while showing the live feed**  **start\_time = time.time()**  **while time.time() - start\_time < 5:**  **frame = picam2.capture\_array()**  **cv2.imshow("Live Camera Feed", frame)**  **# Exits if 'q' is pressed**  **if cv2.waitKey(1) & 0xFF == ord('q'):**  **raise KeyboardInterrupt**  **picam2.stop\_recording()**  **print(f"Video saved as {filename}")**  **time.sleep(0.1)  # Short delay to avoid high CPU usage**  **except KeyboardInterrupt:**  **print("\nExiting...")**  **finally:**  **# Cleans up resources**  **GPIO.cleanup()**  **picam2.close()**  **cv2.destroyAllWindows()** |

Then be sure to run the code using this code:

|  |
| --- |
| **sudo python3 motion\_picamera2.py** |

**To run the mini Illuminated Momentary Pushbutton:**

In order to run the mini Illuminated Momentary Pushbutton, it is required that the button is properly connected to the breadboard and the Raspberry Pi, once it is finished, create a file:

|  |
| --- |
| **nano button\_test.py**  **vim button\_test.py** |

Once the file is open, input this code:

|  |
| --- |
| **import RPi.GPIO as GPIO**  **import time**  **# Set GPIO mode**  **GPIO.setmode(GPIO.BCM)**  **# Define the button pin**  **button\_pin = 4**  **# Set up the button pin as input with an internal pull-up resistor**  **GPIO.setup(button\_pin, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP)**  **try:**  **print("Press the button...")**  **while True:**  **if GPIO.input(button\_pin) == GPIO.LOW: # Button is pressed**  **print("Button Pressed!")**  **time.sleep(0.1) # Debounce delay**  **except KeyboardInterrupt:**  **GPIO.cleanup()**  **print("Exiting...")** |

Run the code:

|  |
| --- |
| **sudo button\_test.py** |

If the button is pressed, it will say “Button Pressed!”.

**To run the code that will transfer the recording to the website:**

In order to run the code that will transfer the recording to the website, it is required that the Raspberry Pi is properly connected to the socket and the website is still active, once it is all done, create a file:

|  |
| --- |
| **nano HomeGuard\_App.py**  **vim HomeGuard\_App.py** |

Once the file is opened, then put in the following code:

|  |
| --- |
| **from picamera2 import Picamera2**  **from picamera2.encoders import H264Encoder**  **from picamera2.outputs import FileOutput**  **from pubnub.pnconfiguration import PNConfiguration**  **from pubnub.pubnub import PubNub**  **import subprocess**  **import time**  **import os**  **# PubNub Configuration**  **pnconfig = PNConfiguration()**  **pnconfig.subscribe\_key = "sub-c-8ed390d9-dba9-407a-b13b-908241df610f"**  **pnconfig.publish\_key = "pub-c-827567a7-63a1-44c0-8e1b-cd2ad828986d"**  **pnconfig.secret\_key = "sec-c-NmM0OTU0MDYtMGE5Zi00YTM0LThiOGEtZjViM2MyMGZlNmVj"**  **pnconfig.uuid = "pi\_motion\_sensor\_1"**  **pubnub = PubNub(pnconfig)**  **# Server Information for SCP Transfer**  **SERVER\_USER = "ubuntu"  # Server username**  **SERVER\_HOST = "52.18.71.193"  # Server IP or hostname**  **SERVER\_PATH = "/var/www/homeguard\_website/static/recordings"  # Server's video directory**  **# Recording Directory on the Pi**  **LOCAL\_PATH = "/home/haroldt2/recordings"**  **if not os.path.exists(LOCAL\_PATH):**  **os.makedirs(LOCAL\_PATH)**  **# Function to Transfer File via SCP**  **def transfer\_file(file\_path):**  **try:**  **print(f"Transferring {file\_path} to server...")**  **scp\_command = [**  **"scp", file\_path, f"{SERVER\_USER}@{SERVER\_HOST}:{SERVER\_PATH}"**  **]**  **subprocess.run(scp\_command, check=True)**  **print("File transferred successfully!")**  **return True**  **except subprocess.CalledProcessError as e:**  **print(f"Error during SCP transfer: {e}")**  **return False**  **# Function to Send Metadata via PubNub**  **def publish\_metadata(filename, timestamp):**  **try:**  **pubnub.publish().channel("motion-detection").message({**  **"event": "Motion Alert",**  **"message": "Motion detected at front door!",**  **"device\_id": "pi\_motion\_sensor\_1",**  **"file\_path": filename,  # Only filename**  **"timestamp": timestamp**  **}).sync()**  **print("Metadata published to PubNub.")**  **except Exception as e:**  **print(f"Error sending metadata to PubNub: {e}")**  **# Main Script**  **try:**  **print("Starting motion detection script...")**  **# Initialize Camera**  **picam2 = Picamera2()**  **video\_config = picam2.create\_video\_configuration(main={"size": (640, 480)})**  **picam2.configure(video\_config)**  **while True:**  **# Generate file names**  **timestamp = time.strftime("%Y%m%d\_%H%M%S")**  **h264\_file = os.path.join(LOCAL\_PATH, f"motion\_{timestamp}.h264")**  **mp4\_file = os.path.join(LOCAL\_PATH, f"motion\_{timestamp}.mp4")**  **# Simulate motion detection**  **print("Motion detected! Recording video...")**  **encoder = H264Encoder(bitrate=1000000)**  **output = FileOutput(h264\_file)**  **try:**  **# Start Recording**  **picam2.start\_recording(encoder, output)**  **time.sleep(5)  # Record for 5 seconds**  **picam2.stop\_recording()**  **print(f"Recording saved: {h264\_file}")**  **except Exception as e:**  **print(f"Error during recording: {e}")**  **continue**  **# Convert H.264 to MP4**  **print("Converting video to MP4 format...")**  **try:**  **subprocess.run([**  **"ffmpeg", "-i", h264\_file, "-c:v", "copy", mp4\_file**  **], check=True)**  **os.remove(h264\_file)  # Delete the H.264 file**  **print(f"Conversion complete: {mp4\_file}")**  **except subprocess.CalledProcessError as e:**  **print(f"FFmpeg conversion failed: {e}")**  **continue**  **# Transfer File to Server**  **if transfer\_file(mp4\_file):**  **# Publish Metadata to PubNub after successful transfer**  **filename = os.path.basename(mp4\_file)  # Only send the filename**  **publish\_metadata(filename, time.strftime("%Y-%m-%d %H:%M:%S"))**  **# Clean up local MP4 file**  **os.remove(mp4\_file)**  **print("Local MP4 file deleted after successful transfer.")**  **else:**  **print("File transfer failed. Skipping metadata publishing.")**  **# Delay before checking for motion again**  **time.sleep(10)**  **except KeyboardInterrupt:**  **print("Script interrupted. Exiting...")**  **finally:**  **picam2.close()**  **print("Camera closed.")** |

The code above will do the simple motion detection, once the motion is detected, it will record the video, save the video as the mp4, transfer the video to the website server, and then afterwards will be viewable on the website.

To run the HomeGuard app:

|  |
| --- |
| **sudo HomeGuard\_App.py** |