Daniel Diamont

EE 445L

4/6/18

**LAB 8 REPORT**

**OBJECTIVES**

Write low-level software drivers for the final project.

Requirements Document: Lab 8

**1. Overview**

**1.1. Objectives:** Why are we doing this project? What is the purpose?

The objectives of this project are to design, build, and test a useful embedded system. Educationally, students are learning about designing Printed Circuit Boards, interfacing hardware and software for a functional embedded system, and mechanical enclosures for an embedded system. My goal is to create an autonomous vehicle that uses computer vision and a discrete PID control algorithm to guide itself through an obstacle course.

**1.2. Roles and Responsibilities:** Who will do what? Who are the clients?

EE445L students are the engineers and the TA, Professor Valvano, and the competition judges are the clients. All tasks will be performed by Daniel Diamont.

* Tasks:
  + Requirements Document
  + PCB Schematic
  + Bill of Materials
  + PCB Design
  + Drivers for hardware interfacing
  + PID controller software
  + Computer-Vision software
  + Enclosure Design
  + Fully functional embedded system

**1.4. Interactions with Existing Systems:** Include this if you are connecting to another board

The system will use a Raspberry Pi 2 Model B board to perform the video-processing, PID control, and DC motor control. The camera will be connected directly to the Pi’s Camera Interface module. Additionally, the system will use a printed circuit board – designed by the student – to interface the Raspberry Pi with DC motors, and an accelerometer + gyro. The system will be powered by a +9.0V battery, and a Linear Dropout Regulator will be used to step down the voltage to +5V for the Pi and the peripherals.

**2. Function Description**

**2.1. Functionality:** What will the system do precisely?

The robot car will interface over Wi-Fi to a Python application on a PC, which will either send a command enabling full autonomy, or manually override directional control of the DC motors for demonstration purposes. Fully-autonomous mode will allow the robot car to capture video of its immediate forward area, process the video to recognize markings to follow and obstacles to navigate, and plot and execute a path through the course using a software PID controller.

The robot car’s software will use multithreading, parallel programming, and interrupt-driven concurrent programming techniques to handle Wi-Fi communication, video-processing, PID control, motor control, and sensor data acquisition at the same time.

**2.4. Performance:** Define the measures and describe how they will be determined.

The system will be judged qualitatively by 9 measures. First, the software modules must be easy to understand and well-organized. Second, the system must satisfy the two input/two output requirements and the multiple ISR requirement. Third, the turnover rate between frame capture and motor control output must be real-time. Fourth, the system must be able to reliably communicate from the controller PC with minimal TCP packet loss. Fifth, the time to run one execution of all threads except for the main thread must be minimized to reduce overhead in the real-time system. Sixth, the design will be judged on whether the robot-car and embedded system enclosure are well-thought out and functional. Seventh, the PCB design must be simple and elegant. Eight, there must be a thorough Bill of Materials to calculate the final cost of the project. Ninth, the system must have mechanisms in place to make it simple to debug the hardware and software in a modular fashion.

**2.5. Usability:** Describe the interfaces. Be quantitative if possible.

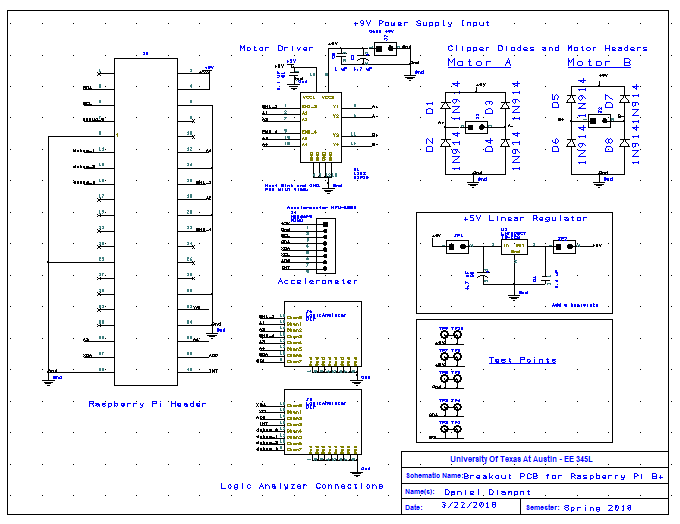
I will use a Raspberry Pi 2 Model B. The Pi will be interfaced with an 5 megapixel ArduCAM OV5647 camera module for the video-capture, an 802.11(b/g/n) Wi-Fi USB adapter for communication, the Raspbian Operating System, Python 2.7 libraries like RPi.GPIO and smbus for low-level drivers, OpenCV2 libraries for computer vision, PID, and motor control, L293 IC for the motor driver, and MPU 6050 for the gyro + accelerometer.

**3. Deliverables**

**3.1. Reports:** Reports for Labs 7, 8, and 11 will be written

**HARDWARE DESIGN**

* See attached UTX-2018S102.sch file.



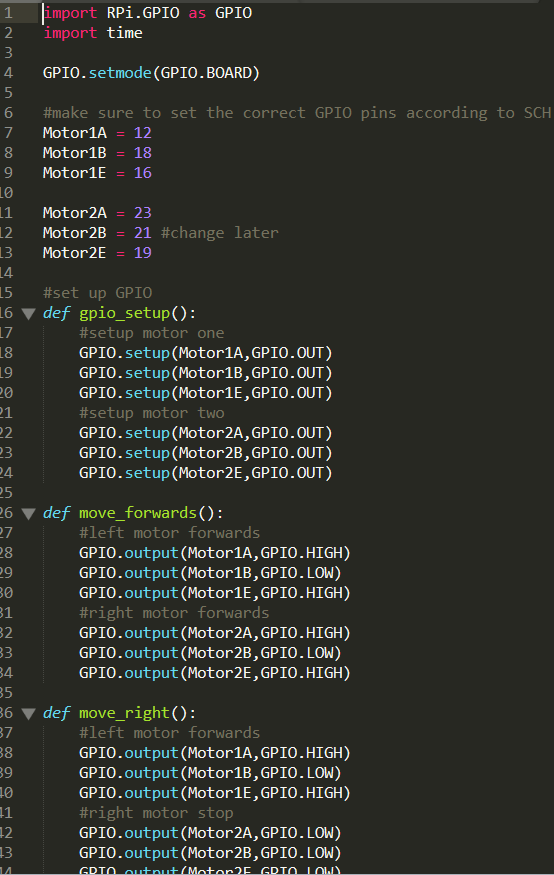
**SOFTWARE DESIGN**

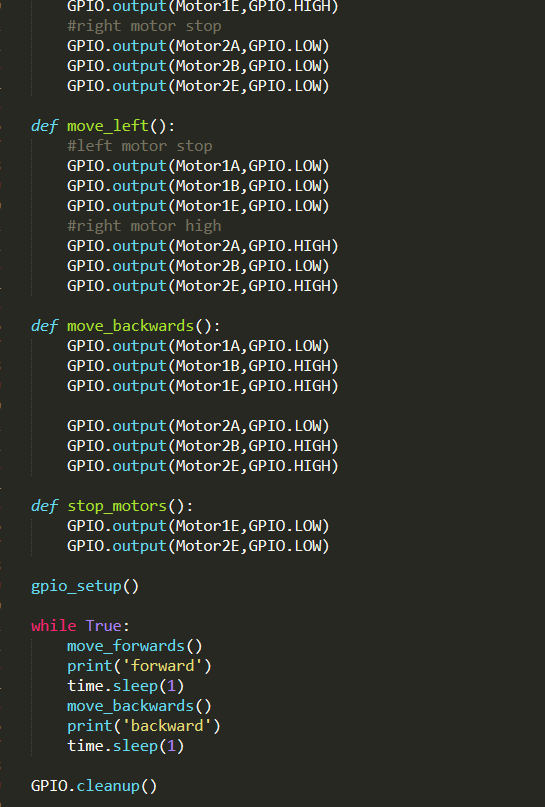
* See attached \*.py files in /drivers/

**Video Driver**

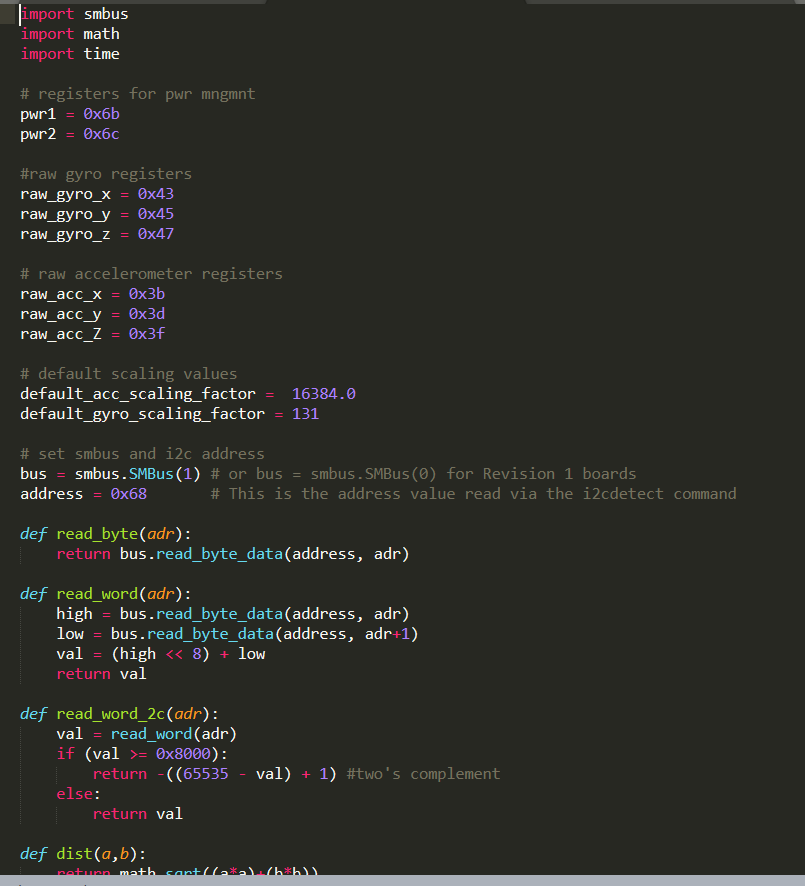


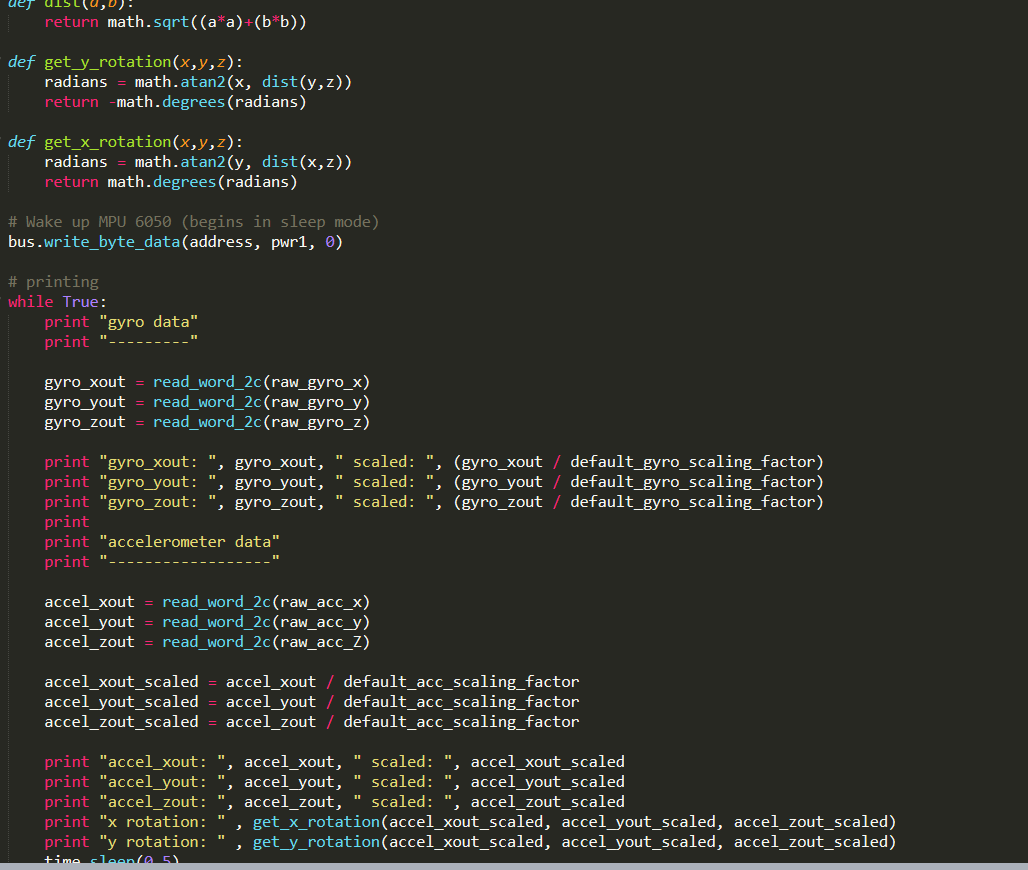
**Motor Controller Driver**





**Accelerometer Driver**

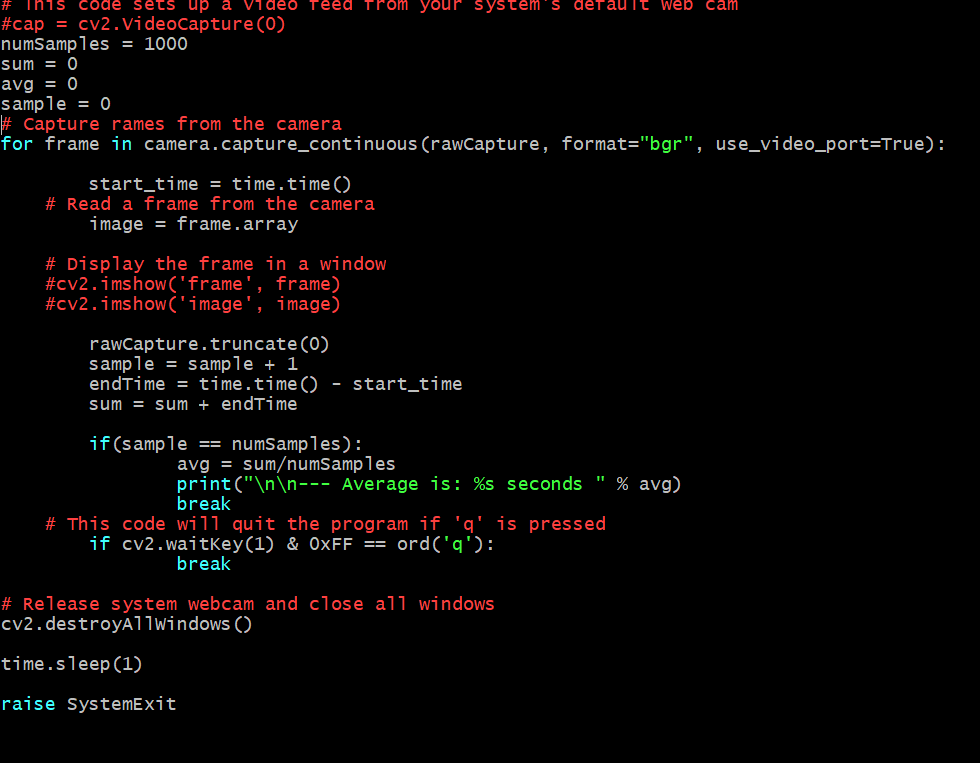


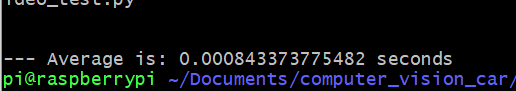


**MEASUREMENT DATA**

**Video Test:**

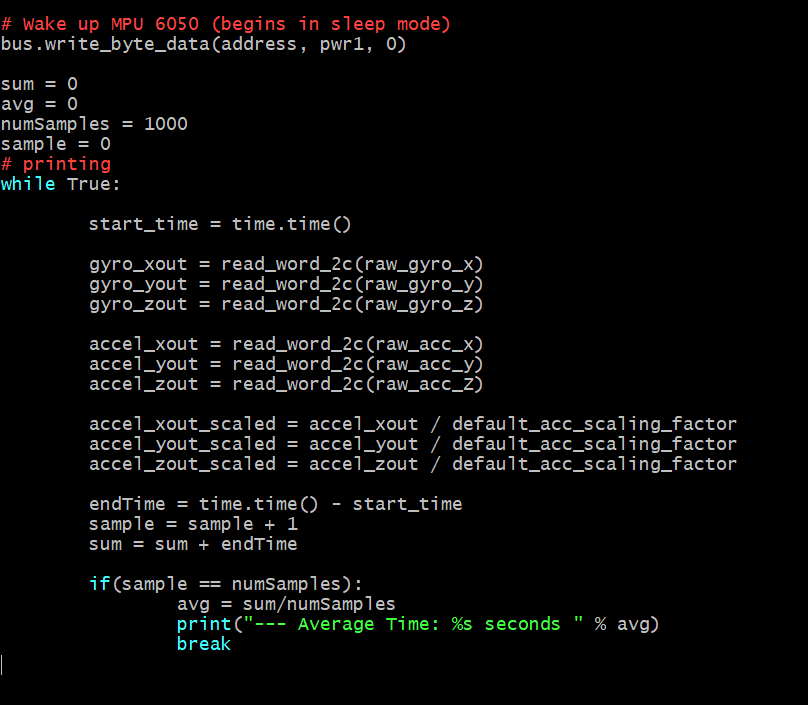
* Parameters:
  + 30 frames per second
  + Resolution: 480 x 320
  + Capture Format: BGR
  + **1000 Trial Average Thread Runtime: 843.373 us**





**Accelerometer Test:**

* Parameters:
  + I2C default speed of 100 kbps
  + MPU – 6050 Accelerometer + Gyro
  + Computation of scaled values for all 6 axes
  + **1000 Trial Average Thread Runtime: 6.051 ms**





**Observations:**

It is still necessary to compute overhead for the PID controller thread, and for the computer vision data extraction subroutines to provide a rough estimate of the computational load distribution of the system.

Furthermore, it is worthy to note that the speed of the I2C bus can be increased four-fold to 400 kpbs for a possible added improvement of 2 ms in data acquisition overhead. It is important to consider adding a software circular buffer for acquiring data from the accelerometer + gyro, or simply to consider using a low polling frequency to further decrease the computational load on the BCM2835 processor.