               I have created a robot controlled with a joystick using the Raspberry Pi.  This project has taught me numerous concepts and tidbits.  Between interfacing the Raspberry Pi with the motors and joystick and actually programming the Raspberry Pi, there have been many struggles and successes.  The Pi is a great candidate for controlling circuits as it has many pins that can be controlled via an easy to learn programming language—python.

           I used a joystick, analog to digital converter, motor controller, and two motors along with the Raspberry Pi.  Because the Pi only supports digital signal—which is only on or off—and the joystick outputs an analog signal—which can vary based on the reading—I needed to use an analog to digital converter (ADC).  This converter works through serial peripheral interface (SPI).  Essentially the chip has 3 pins connected to the Pi.  One pin for the chip to send signals to the Pi, one for the Pi to send signals to the chip, and the last pin is the clock which regulates when the signals are sent.  Since all communication is done through a one way connection pin, all data received and sent must be binary.  In order to receive a bit, a bit must be sent.  This is how the Raspberry Pi converts analog to digital through the use of a chip (I used the MCP3008 specifically).  The Raspberry Pi controls the motors through the use of a motor controller.  Essentially there are three pins connected for each motor, one activation pin and two controlling pins.  The Raspberry Pi is in the middle of the ADC and motor controller.  The Pi interprets the ADC signals and sends out the correct signals to the motor controller.  I also had an LED blinking to show that the program was running.

           My initial plan was to control the Raspberry Pi through using another computer that was connected via Wi-Fi.  This proved to be a problem because of the way the Pi performed pulse width modulation (PWM).  Since the Raspberry Pi is completely digital, it can only send out either on or off and no in between.  The way the Pi performs analog signals is through PWM.  But the Raspberry Pi does not natively support PWM.  Instead python simulates this through the use of a continuous background while loop that essentially turns the signal on for a little bit, then off, then back on, and repeats this for the time PWM is on.  Because this is done through software, when the program is terminated the PWM signal also stops.  In order for me to try to get around that, I would have to have the program run in the background and stop it when I would change the value the PWM signals would run at and stop the other running program.  That solution would be too complex for the scope of this project.  Instead I changed to controlling the robot via a joystick.  This allowed me to keep the program running and thus the PWM would continue to run.

           The software used to control the robot is written in python.  Essentially the whole program is a while loop which reads the data from the joystick, converts it to the correct values for PWM, and sets the PWM and repeats.  Because the Raspberry Pi is an actual computer, the program needed to be started by the system at startup.  To accomplish this, I just added a command in the startup files.

           I learned many valuable lessons on this project from the wiring to the programming.  Even though I did not implement the project the way I had hoped, I still think the project was successful. The whole project can be found at <https://github.com/rosskyl/raspberry-pi-robot>.