ADAPT

Autonomous **D**riving for **A**dverse **P**erceived **T**errain

ILR01

By: Bryson Jones

Team Shasa Antao

Shaun Liu

Evan Schindewolf

Wesley Wang

Team B

Master of Science in Robotic Systems Development
Robotics Institute
School of Computer Science
Carnegie Mellon University

February 12, 2019

Individual Progress

Sensors and Motors Lab

During this lab, my main role was to develop code and design a circuit that included an IR proximity sensor, servo motor, and a push button for switching between states. I also initialized and oversaw the revision control repository that the team utilized to store and merge code together, along with documenting pin use and circuit function within a README file. I also assisted with laying out and assembling the final circuit onto our demonstration platform, and debugging the final code.

To write and test code for the IR proximity sensor, switch-button, and servo motor control, I built the proto-typing circuit seen in figure 1 below. This circuit supplies the sensor signal to the A0 pin, the push button signal to digital 2 pin, and the servo position is commanded from digital 9 pin, on the Arduino Uno board.

The code implemented to run on this circuit functions in two modes, which can be switched between by pressing the button included on the breadboard. The first is where the position of the servo motor is controlled by the distance reading of the SHARP IR proximity sensor, which has the reading bounded between 3 and 24 inches. The second is having the servo position specified by user input, accepted in degrees.

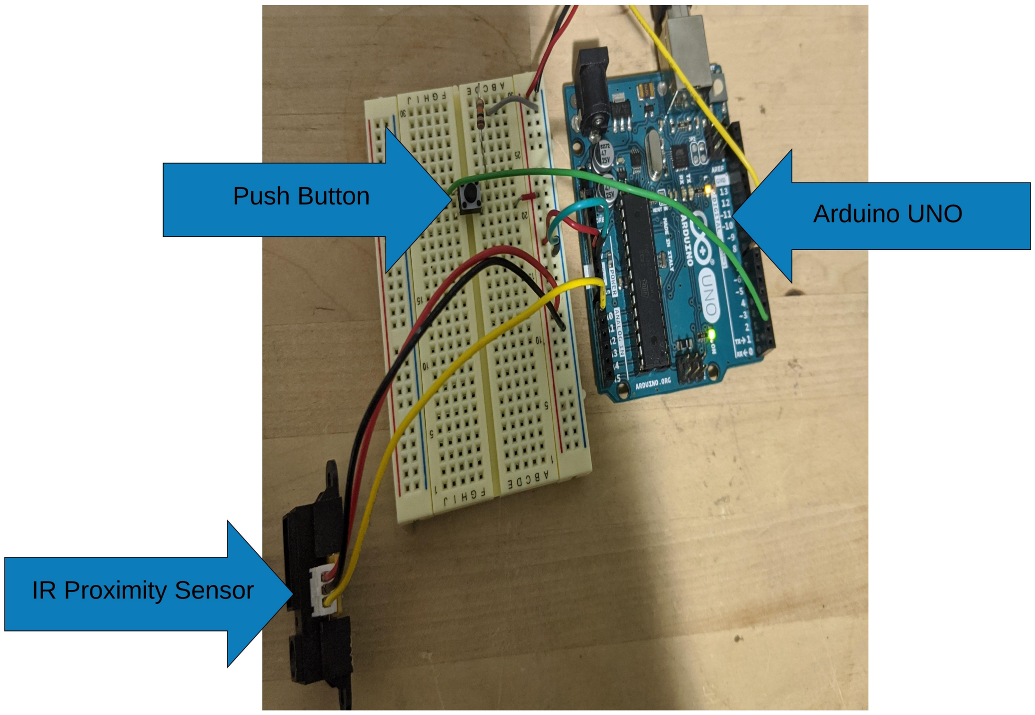

Figure 1: Prototyping Circuit

IR Proximity Sensor

The SHARP IR Proximity sensor that our team utilized within this lab is an analog sensor, which provides a signal reading with 10 bit precision, where I found the maximum reading to be ˜680. To effectively use the sensor, it was first necessary to apply a filter to the raw signal to remove noise, and I decided to implement a moving average filter. I was initially concerned that this could cause an issue with signal lag, but after tuning the window, I was able to find a good middle ground between a slow response and noisy signal. Then, to understand the sensor readings, I needed to map the sensor output to a physical value of distance, for which I referred to the datasheet and found a plot that displays the relationship between output voltage and distance, which can be seen in Figure 2.

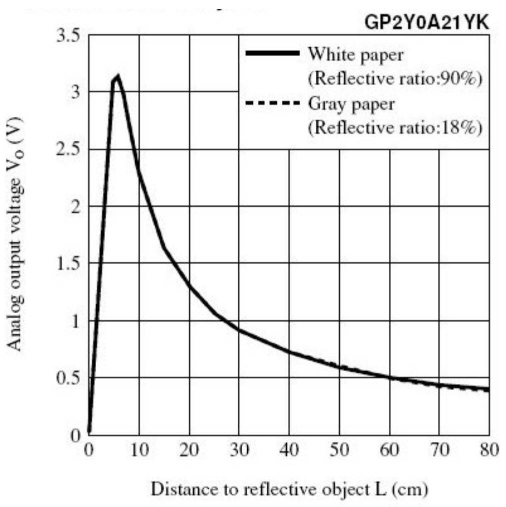

Figure 2: SHARP IR[1]

Seeing this plot, it is obvious that the relationship is extremely non-linear throughout the operating range of the sensor. Being aware that this is a common hobbyist sensor, I researched for functions that had been created to map the sensor readings to a usable distance value, and found an equation relating the 10 binary bit representation to a distance in inches, for this specific model of IR proximity sensor [2]. I implemented this equation within my code, and performed checks at a few distances with a measuring tape to ensure that this conversion function was correct. I noted that the conversion appeared very accurate up to about 24 inches away from the sensor, and based on this, and minimum distance ratings on the datasheet, I decided to constrain the output range of the distance between 3 and 24 inches.

Servo Motor

Interfacing with the servo motor was performed through a built-in library within Arduino, called Servo. This allowed me to attach a pin on the Arduino board that would be reserved for commanding the servo position. The motor requires 5V of power, so from there I connected the 5V and ground wires of the motor to their respective locations on the breadboard to test, where the voltage was supplied by the Arduino. After briefly testing, I setup a separate supply of voltage to the breadboard from our team station power supply, to ensure that the servo had sufficient power being provided. I then mapped the distance reading of the sensor to integer values between 0 and 180, which would be fed into the servo motors as a position value in degrees.

Push Button

The push button feature that I added to the circuit was used to switch the system state from being controlled by sensor feedback to accepting user inputs from our GUI. The state change occurs within an attached interrupt function, where the button signal is debounced by specifying that if the signal changes more than once within 300ms, it is a bounce, and the state change should be ignored.

Full Circuit

The full circuit assembled and tested by all team members can be seen below, in Figure 3:

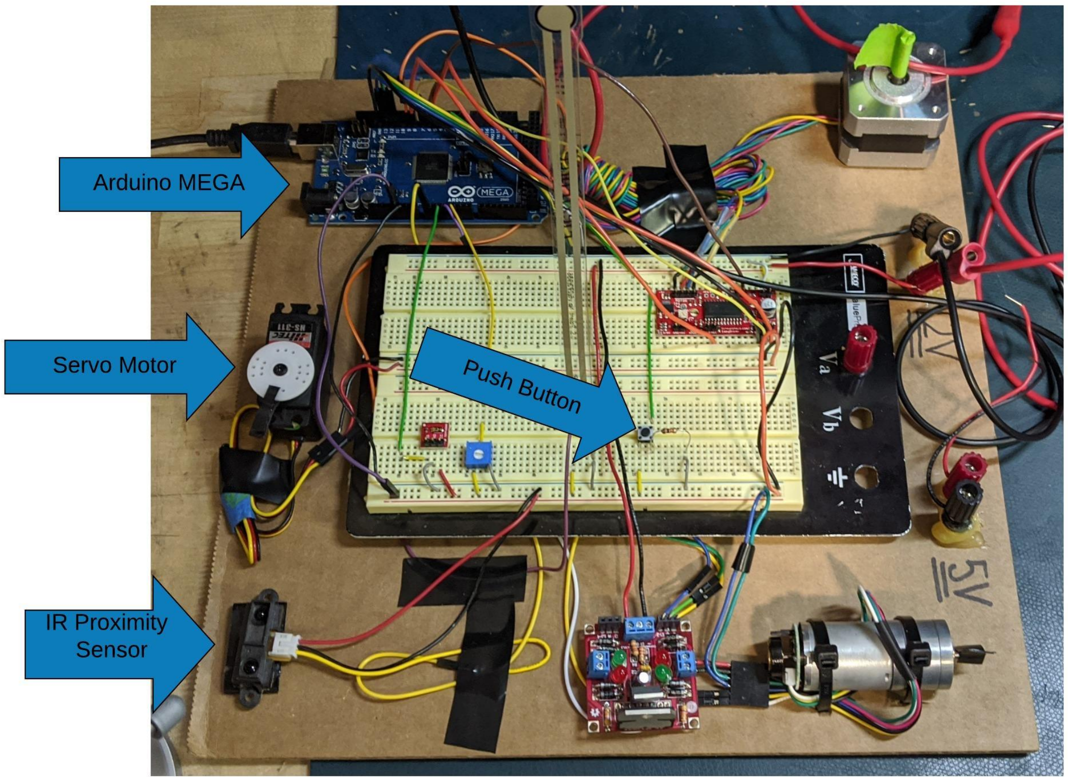

Figure 3: Fully Integrated Circuit

ADAPT System

My focus related to the development of the ADAPT system so far has been broken into two main categories. The first is the planning and developing of the dynamic simulation environment. I initially started the semester out by researching dynamic vehicle models capable of handling non-linear tire dynamics, and deciding on a simulation environment. We have decided to go forth with a dynamic bicycle model and brush tire model, and have been coding up our initial simulation script within MATLAB. The second is the sensor selection and hardware design, where I have mainly focused on developing our CAD model for the vehicle, compute box housing, and all fastening methods, and researching and testing sensors for the vehicle.

Challenges

Sensors and Motors Lab

The main challenge we faced during this lab was in the final integration, and subsequent interfacing with our ROS GUI. We initially ran into an issue with insufficient dynamic memory being available on our Arduino UNO unit, which was caused by the large allocation that is required for ROS to communicate with serial. We solved this by sourcing an Arduino MEGA unit, which had 4 times the dynamic memory capabilities as the UNO unit. With that problem solved, we quickly ran into another, which was a communications issue between ROS and the Arduino, that we attributed to an incorrect ordering of commands within the script to initialize ROS.

While our team made efforts to document code and pin use for others to refer to before performing their own work, we still rain into issues getting the overall code to run after compiling all of the separate pieces together. Some of this ended up being related to hardware assembly issues, and some attributed to poorly named variables. We next went through the board connections row by row to ensure that each sensor, motor, or other external components had the correct ground and power, and while doing this, we found that our stepper motor driver board was damaged during the transition to the full circuit, causing us to have to replace it.

ADAPT System

A main hold up that was unforeseen for the project work is related to setting up the NVIDIA Jetson Xavier computing unit with the correct software to run and test algorithms for our terrain comprehension pipeline. This is related to many of the popular frameworks such as TensorFlow or PyTorch not releasing steady version for ARM architecture computers, which makes it difficult to transfer pre-built models onto the device. After much debugging and testing we have been successful in loading and testing a few different algorithms, but this did waste some time that was supposed to be dedicated to model development. This has caused a bit of delay in our timeline, which has forced us to start thinking of ways to condense other aspects to get back on track.

Teamwork

To split the work amongst the team equally, we separated individual hardware and software aspects up to distribute. The following outlines what each team member focused on for this lab

Shasa Antao

Shasa worked on setting up the circuit and implementing the PID controller for position and velocity of the DC motor. This was likely the most challenging of the motor control setups to implement.

Shaun Liu

Shaun worked on writing code for the ambient light and thermistor sensors for our team, along with assisting in debugging the final circuit.

Evan Schindewolf

Evan worked on developing the GUI for the lab. We decided that it would be useful for the GUI to be architected in ROS, to allow the team to get more familiar with ROS tools and abilities since we are planning to have ROS play a communications role within our MRSD project. This was a task that we thought might take a substantial amount of time, since none of the team was previously very familiar with utilizing ROS.

Wesley Wang

Wesley worked on writing code and designing the circuit to read in sensor responses from the flex-force sensor, and interface that with the stepper motor.

Future Work

Looking forward for the project, my main focus is to continue development of the dynamic simulator, and transition to prototyping various control algorithms. We have moved a bit slower in this process than we originally wanted to, but have laid out a solid road map for work to occur going forward. The main person I am working with to develop this code is Evan, whom I sat down with the document goals and best practices when developing the simulator. We are emphasizing writing the code in such a fashion that it is relatively easy to transfer it to our robot so that we can gather simulation and real-world data for a scenario, and then overlay them to compare control responses. On top of this, I will be working to assembly all of our manufactured and ordered parts onto our platform, to ensure that it is ready for preliminary testing.