OuterSense

Automated Driving Using External Perception

Teammates: Ronit Hire, Dhanesh Pamnani, Shreyas Jha, Jash Shah

ILR01

Feb 9th, 2023

Sponsor: Nissan Automotive Inc.
Liam Pederson, Najam Baig, & Viju James

THE ROBOTICS INSTITUTE Carnegie Mellon University

Contents

1. Individual Progress

1.1 Sensor Motor Lab

My responsibility for the sensor motor lab was to integrate the temperature sensor with the RC servo motor. Additionally, I was also responsible to develop and implement the entire electrical connection for the integrated circuit consisting of RC servo motor, DC motor, stepper motor, IR sensor, temperature sensor, ultrasonic sensor and the Arduino Mega controller along with other necessary components like button, resistors, capacitor and drivers.

Servo motor: The motor has 3 connections – ground, power and signal. Connect the power to 5v, ground with controller ground and signal to digital output pin on the controller. To program the servo motor existing servo library is used and digital pin is configured to work as output of the controller to send commands to servo. Position commands are written such that motor reaches desired position

Temperature sensor: The sensor has 3 connections – ground, power and signal. The ground is connected with controller ground, 5V is given as power and signal is given to analog pin of the controller. The voltage values read on the analog pin are passed through a moving average filter and then converted to appropriate value multiplying by a factor to get corresponding temperature readings.

Integrate servo and temperature sensor: The servo will stop moving if the temperature goes beyond a certain value for example 80 deg F.

Electrical connections: All the components are connected correctly to get desired output from each sensor and motors. All the wires are tied, properly routed and components are secured. Refer figure 1.1 for final circuit.

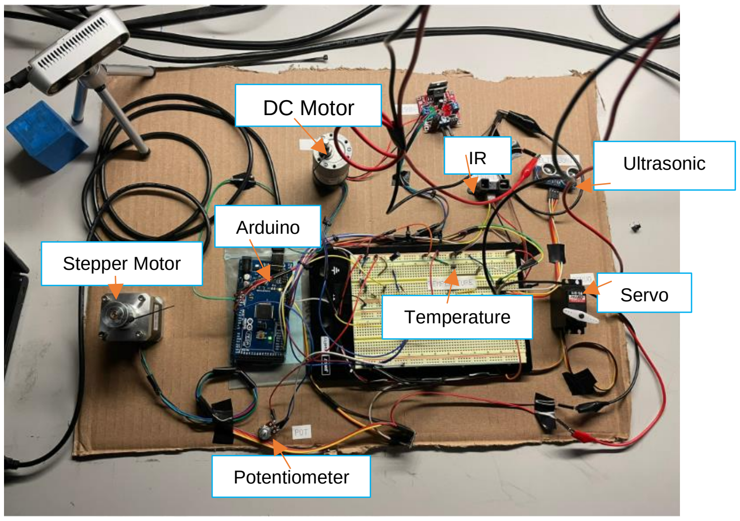
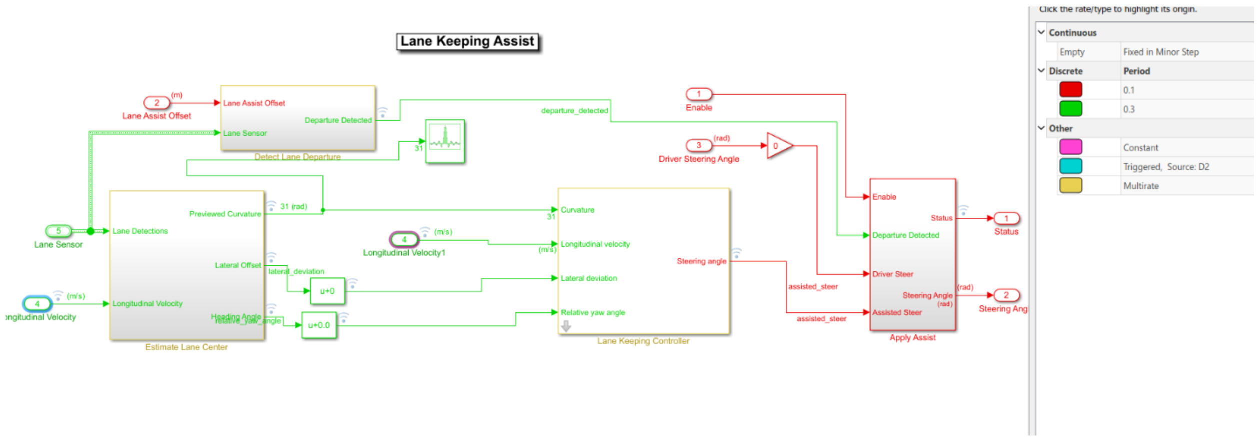


Fig 1.1 : Fully Integrated Circuit

1.2 MRSD Project: Team OuterSense

My focus related to development of the MRSD project so far has been towards the control system architecture. I started off the semester by researching control algorithms to keep the RC vehicle in the lane while it is following the desired trajectory. I finalized on Model Predictive control approach for implementing lane keeping for the vehicle. I did a thorough study to understand the working and architecture of MPC before implementing. I studied research papers and articles for MPC implementation along with watching series of videos on MPC design and implementation.
I started by implementing MPC for lane keep on Simulink model. To get a better understanding I designed a dummy track on Driving scenario designer on MATLAB and analyzed the behavior of the vehicle on the track. For the vehicle to stay within the lane the MPC needs two inputs mainly the lateral offset of the vehicle from the center line and the deviation in yaw of the vehicle with respect to the center line. These two inputs are calculated based on the sensor data collected. The goal of the project is to use external perception, since the sensor is not mounted on the vehicle there will be some offset and delay in computing the sensor data. Hence, one of the objectives in the simulated environment was to simulate the MPC behavior when there is an offset in lateral deviation and yaw angle measured from the sensor and also observe the effect when there is a delay in sending/receiving data from MPC controller. Certain blocks on Simulink were added to generate offset/ noise on data to simulate errors in sensor reading, a delay block was added and the loop rate (sample time) of MPC controller was varied to simulate delay in MPC computation. All the results were gathered and analyzed. Refer figure 1.2 for Simulink model.

Currently I am working on writing the MPC for lane keeping in C++/python which can then be integrated with hardware.



Click the rate/type to highlight its origin.

Continuous	
Empty	Fixed in Minor Step
Discrete	**Period**
🟥	0.1
🟩	0.3
Other	
🟪	Constant
🟦	Triggered, Source: D2
🟨	Multirate

Fig 1.2 : Simulink model

2. Challenges

2.1 Sensor and Motor Lab

One of the challenges I faced was with the DC motor and its driver. Initially we were not getting accurate values from the DC motor encoder, to solve that I added pull up resisters to the two encoder signal that were given to the controller. We then got desired results.

After few operations I faced issue with the DC motor driver, the driver heated up. On spending few hours on debugging the issue I decided to go ahead with a new motor driver. I had to solder all the components on the PCB to build the new motor driver. After few operations I start to face the same issue but this time I was able to narrow down the problem and identified that we had a faulty DC motor. We got desired results on changing the DC motor.

2.2 MRSD project

The main challenge I faced was understand the MPC block on MATLAB. It was difficult to understand how it was implemented and how the output was affected on changing few design parameters. I watched Mathwork video series on MPC and on Simulink to get comfortable with MPC controller on Simulink. I also studied all the Simulink examples on MPC implementation.
Another challenge was to simulate the two scenarios – having offset in sensor data and delay in computation. After lot of researching, I added noise and delay blocks but I was not getting satisfactory results. I consulted with my MRSD seniors on how to approach this problem and then implemented constant bias to sensor readings to generate offset and changed the sample time of MPC controller to simulate the delay in MPC computation.

3. Team Work

Ronit Hire: Ronit had software experience and had worked on GUI before and hence the responsivity was developing the GUI was given to him. When all the code was integrated and the fully integrated circuit was connected, I sat with Ronit to test the final code on the circuit. Towards development of MRSD project Ronit is working on the perception front, the work which he is implementing will be given as input to my control system architecture. He is also handling the logistics for electronic and mechanical components.

Shreyas Jha: Shreyas worked on the stepper motor and potentiometer sensor. He also worked on IMU sensor which is required for our MRSD project where we have to read and analyze the IMU sensor readings mounted on the RC car. Shreyas also worked on integrating the individual codes for all the motors and sensors into one integrated code to interface with the GUI. From a project point of view He is currently testing all the components that need to me mounted on the RC vehicle and work towards building the RC car.

Dhanesh Pamnani: Dhanesh has implemented the code for push button and debouncing along with writing a program to read data from IR sensor. Additionally, he is also plotted the transfer function for IR sensor. From project point of view he is in charge of the track, he has designed the track on CAD and at the moment is building the track in the machine shop. He is also handling the logistics associated with track items.

Jash Shah: Jash wrote the program for the speed and position control of DC motor. He has also worked on ultrasonic sensor and interfacing the sensor with DC motor. From MRSD project point of view he is working on the perception side where he is currently detecting objects and calculating the orientation of the object.

4. Future Plan

The next step for me towards development of MRSD project is to implement MPC controller in C++ / python as well as test the controller behavior. I also plan to help in testing components of RC car and assemble it.

5. Quiz

1. Reading a datasheet. Refer to the ADXL335 accelerometer datasheet (https://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf) to answer the below questions.
   - **What is the sensor's range?**
     Ans: ±3.6 g

   - **What is the sensor's dynamic range?**
     Ans: 7.2 g

   - **What is the purpose of the capacitor C_{DC} on the LHS of the functional block diagram on p. 1? How does it achieve this?**
     Ans: A single 0.1 μF capacitor, CDC, placed close to the ADXL335 supply pins adequately decouples the accelerometer from noise on the power supply
     Adding a capacitor between the power supply and ground in a sensor circuit helps to reduce supply noise. This noise can interfere with the accuracy and stability of the sensor measurements. The capacitor acts as a low-pass filter and helps to smooth out any voltage fluctuations.

   - **Write an equation for the sensor's transfer function.**
     Vout = Sensitivity * (Acceleration + Offset
     Vout = 0.3*(acceleration) + 1.5

   - **What is the largest expected nonlinearity error in g?**
     Ans: 0.3% of 7.2g = 0.0216g

   - **What is the sensor's bandwidth for the X- and Y-axes?**
     Ans: 1600 Hz

   - **How much noise do you expect in the X- and Y-axis sensor signals when your measurement bandwidth is 25 Hz?**
     Ans: 150 * sqrt(25 * 1.6) = 945 μg

   - **If you didn't have the datasheet, how would you determine the RMS noise experimentally? State any assumptions and list the steps you would take.**
     Record multiple samples of the sensor signal in a stationary environment with no external acceleration. This will ensure that the readings are only due to noise and not due to any external signals.

Subtract the mean value of the samples from each sample to remove any DC offset. This will help to focus on the noise component of the signal.

Square each sample and average the squared values. This will give you the mean squared value of the noise.

Take the square root of the mean squared value to obtain the RMS value of the noise. This is the RMS noise of the sensor signal.

2. Signal conditioning
   - Filtering
     - **Name at least two problems you might have in using a moving average filter.**
       -A moving average filter can smooth out the input signal, which can result in a loss of information or a reduction in the resolution of the signal
       - A moving average filter can introduce a lag in the output signal, as it requires a certain number of samples to produce an average. This can result in a slow response to rapid changes in the input signal

     - **Name at least two problems you might have in using a median filter.**
       -The choice of window size for a median filter can have a significant impact on its performance, as a larger window size will result in more smoothing and a smaller window size will result in less smoothing**.**
       - It is computationally complex as they require sorting of values and selecting a median value which can be time consuming.

   - Opamps
     - In the following questions, you want to calibrate a linear sensor using the circuit in Fig. 1 so that its output range is 0 to 5V. Identify in each case: 1) which of V1 and V2 will be the input voltage and which the reference voltage; 2) the values of the ratio Rf/Ri and the reference voltage. If the calibration can't be done with this circuit, explain why.
       - **Your uncalibrated sensor has a range of -1.5 to 1.0V (-1.5V should give a 0V output and 1.0V should give a 5V output).**
         V2 should be input voltage and V1 should be reference voltage at -3v
         Vout = (V2-V1)Rf/Ri + V2
         0 = (-1.5 – V1) Rf/Ri – 1.5
         -(1.5 *(Ri/Rf) + 1.5) = V1…………..i

         5 = (1- V1) Rf/Ri + 1
         -(4 (Ri/Rf) -1) = V1…………………ii
         On solving i and ii we get

Rf/Ri = 1 and V1 = -3

- **Your uncalibrated sensor has a range of -2.5 to 2.5V (-2.5V should give a 0V output and 2.5V should give a 5V output).**
  Vout = (V2-V1)Rf/Ri + V2
  0 = (V2 +2.5)Rf/Ri +V2
  V2 = (V2 + 2.5) * Rf/Ri……..i

  5 = (V2 – 2.5)Rf/Ri + V2
  5 = (V2 – 2.5) * V2 / (V2 + 2.5) + V2
  On solving
  V2 = 4.04
  And Rf/Ri = 0.6177
  V1 will be input voltage and V2 is reference voltage

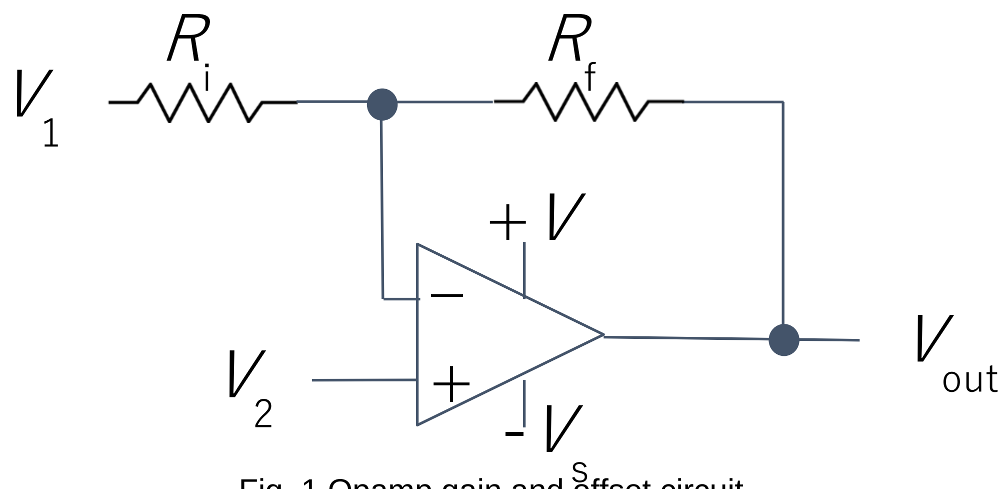


Fig. 1 Opamp gain and offset circuit

3. Control
   - **If you want to control a DC motor to go to a desired position, describe how to form a digital input for each of the PID (Proportional, Integral, Derivative) terms**.

     Ans: The PID terms can be formed as digital inputs to the control system as follows:
     Proportional (P) term: This term provides a control action proportional to the error between the desired position and the actual position of the motor. The digital input for this term can be generated by subtracting the desired position from the actual position and multiplying the result by a proportional gain (Kp).

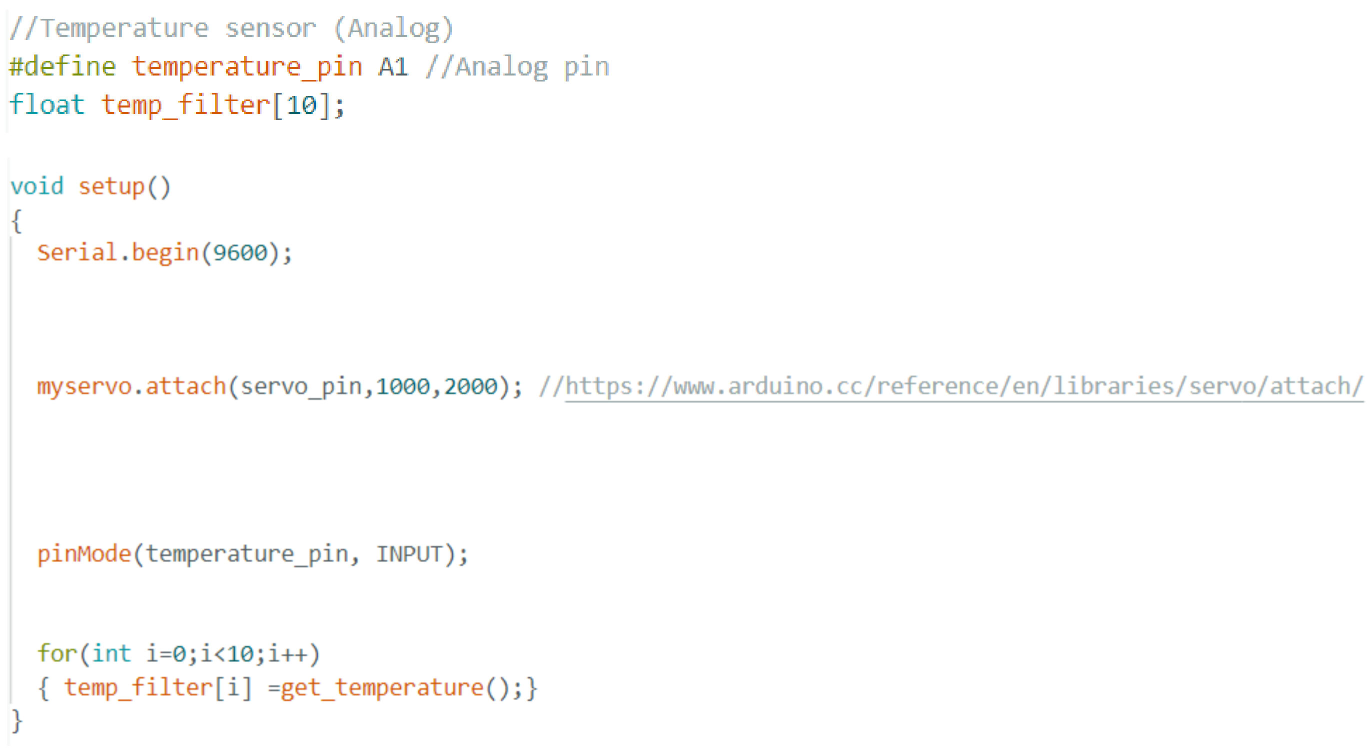
Integral (I) term: This term provides a control action proportional to the accumulated error over time. The digital input for this term can be generated by integrating the error over time and multiplying the result by an integral gain (Ki). Derivative (D) term: This term provides a control action proportional to the rate of change of the error. The digital input for this term can be generated by differentiating the error and multiplying the result by a derivative gain (Kd).

- **If the system you want to control is sluggish, which PID term(s) will you use and why?**
  If the system is sluggish, increasing the Proportional term will increase the control action and speed up the response of the system.

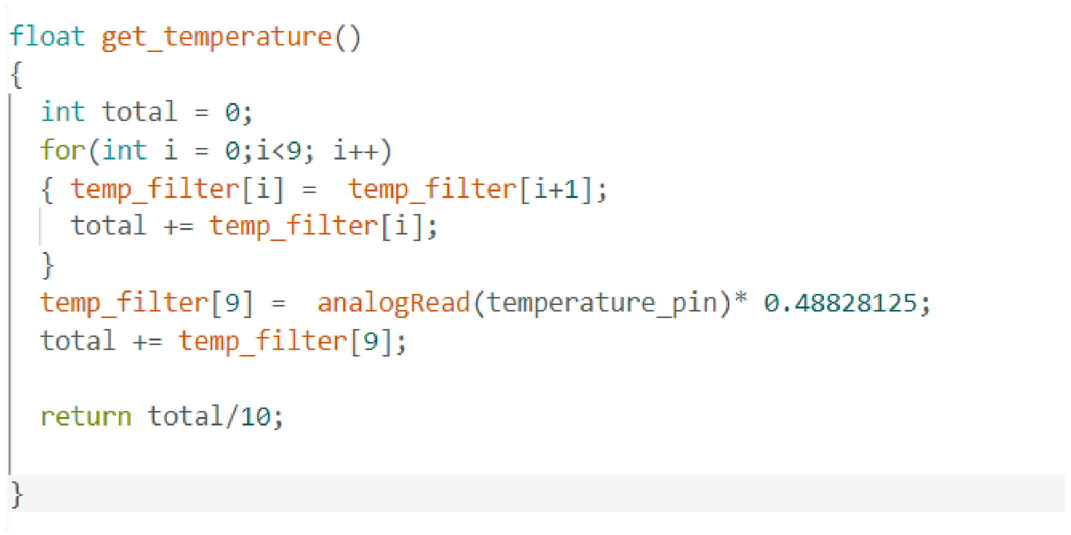
- **After applying the control in the previous question, if the system still has significant steady-state error, which PID term(s) will you use and why?**
  Ans: We will adjust the Integral term. The Integral term provides a control action proportional to the accumulated error over time, which can help to eliminate any steady-state error in the system.

- **After applying the control in the previous question, if the system still has overshoot, which PID term(s) will you apply and why?**
  The Derivative term provides a control action proportional to the rate of change of the error, which can help to reduce overshoot and improve the transient response of the system. In a system with overshoot, increasing the Derivative term will help to dampen the response and reduce the overshoot.

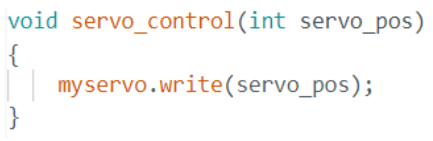
6. Code



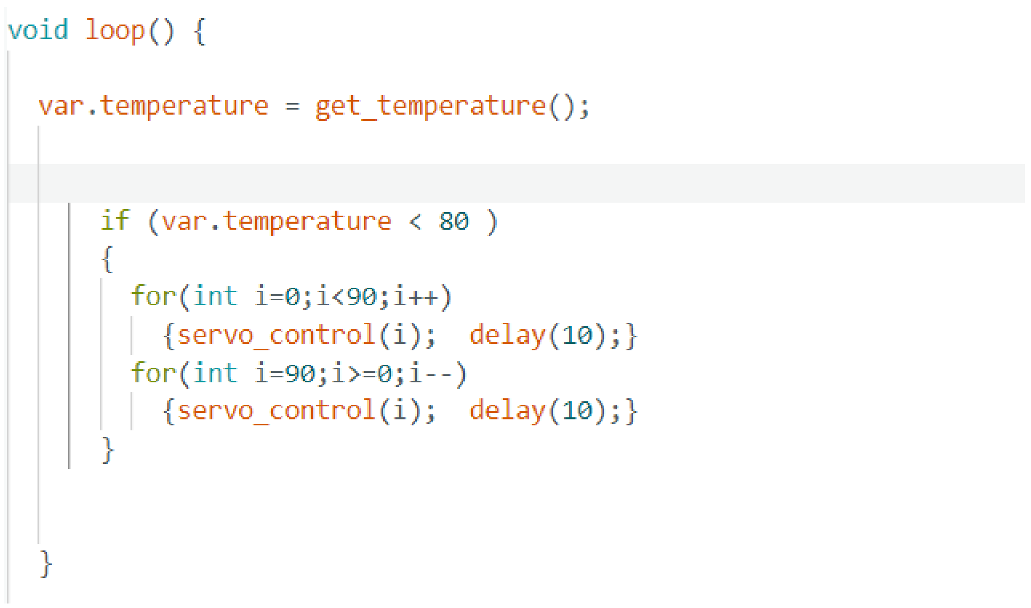
```
//Temperature sensor (Analog)
#define temperature_pin A1 //Analog pin
float temp_filter[10];

void setup()
{
  Serial.begin(9600);



  myservo.attach(servo_pin,1000,2000); //https://www.arduino.cc/reference/en/libraries/servo/attach/




  pinMode(temperature_pin, INPUT);


  for(int i=0;i<10;i++)
  { temp_filter[i] =get_temperature();}
}
```



```
float get_temperature()
{
  int total = 0;
  for(int i = 0;i<9; i++)
  { temp_filter[i] =  temp_filter[i+1];
    total += temp_filter[i];
  }
  temp_filter[9] =  analogRead(temperature_pin)* 0.48828125;
  total += temp_filter[9];

  return total/10;

}
```



```
void servo_control(int servo_pos)
{
    myservo.write(servo_pos);
}
```



```
void loop() {

  var.temperature = get_temperature();


    if (var.temperature < 80 )
    {
      for(int i=0;i<90;i++)
        {servo_control(i);  delay(10);}
      for(int i=90;i>=0;i--)
        {servo_control(i);  delay(10);}
    }


  }
```