William Seto

Team B: Auto Pirates

Teammates: Bikramjot Hanzra, Shiyu Dong, Tae-Hyung Kim, Tushar Chugh

ILR01

October 16, 2015

Individual Progress

For this lab, I was in charge of calibrating the Sharp IR sensor and obtaining accurate distance measurements. Also, I assisted Shiyu with the PID controller for the DC motor. Finally, I integrated the code for the sensor and motor control.

For initial filtering on the IR sensor, I added a 10uF electrolytic capacitor and a 100nF ceramic capacitor near the sensor. Also, since the sensor has a non-linear response, some work must be done to correctly map the ADC readings to distance values. I leveraged an arduino library online, which includes some averaging and thresholding features to provide cleaner values. This proved sufficient, as our sample graphs matched up well with the ones in the datasheet.

For the PID tuning, I aided Shiyu with tuning of the PID constants. Our strategy was to increase Kp until we saw some oscillation, lower Kp, then increase Kd to reduce the oscillation. We experimented with Ki but ended up not using it. We would have steady-state error, but found that if we increased Ki too much, our system would become very unstable. We were satisfied with errors of +- 2 degrees, since our encoder has low resolution anyway.

For the system integration, I integrated all the sensors first, making sure all the readings were correct. For integration of all the motors, I had to create wrapper functions for the functionality, since everyone’s code had been in the typical arduino loop. By creating wrapper functions, the main loop on the integrated code consisted only of serial processing and a switch case for processing each mode of our system. The basic structure of our program is seen below.



*Figure 1: Arduino program flow chart*

Challenges

Difficulties with the IR sensor primarily involved calibration to obtain accurate distance values. I tried to come up with my own linearizing equation, but eventually found that the library was much more useful since it had averaging features in addition to a linearizing equation.

The main difficulty we faced in tuning our PID controller was due to the fact that it is extremely sensitive to the rate at which our control loop runs. We had tuned it well when developing in isolation, but we had to repeat the process again when integrating it with the whole system. We also found that printing a large amount of debugging statements would add delays as well, so we were careful to test the system as if it was demo ready, when the only messages sent are the updates to the GUI.

Integrating the code was mostly painless, but we had to spend a lot of time thinking before-hand on how best to do it. Since we opted not to use timer interrupts to do critical things like read sensor data, send updates, and process updates, it was crucial that our loop ran as fast as possible. As a result, I had to refactor some functionality which was wrapped in a blocking loop. This meant creating more variables to hold the state of some functions, such as PID, or position control on the stepper motor.

Teamwork

Shiyu Dong

Shiyu integrated the force resistive sensor with Arduino and implemented the PID controller for position and velocity control on the DC motor. He worked with William to tune the controller.

Bikramjot Hanzra

Bikram integrated the toggle switch and servo motor with Arduino. He also wrote the matlab code to plot the graph of ADC output vs. distance for the Sharp IR sensor.

Tae-Hyung Kim

Tae-hyung integrated the potentiometer and stepper motor with Arduino. He also aided in general debugging during total integration.

Tushar Chugh

Tushar developed and integrated the GUI with Arduino. He defined the message protocols for the serial communication between Arduino and the computer. In addition, he implemented mode transition when the push button fired.

Plans

From now until the next demo, I plan to focus on project work. As I am on the perception team, my primary goal is to extract usable data from the radar so that we can perform object detection. We have found an open source plugin which interfaces to the radar and displays the radar image on our computers. I will be digging into the plugin source to find where we will make modifications so that we can save the data to a log file. The next task will be to modify the plugin so that it can publish this data to a ROS topic, thereby integrating the radar with the rest of our system.

Code

Here is some code I wrote.

/\*\*\* serial stuff \*\*\*/

char mode = '7';

char dir = '0';

long val = 0;

char type = '0';

int stateChanged = 0;

volatile unsigned long last\_gui\_update\_time = 0;

void loop() {

sendData();

stateChanged = 0;

if (Serial.available() > 0) {

String input = Serial.readString();

if (input[0] == 'A' && input[1] == 'P' && input[8] == 'K') {

//if we got a new command, reset values

resetValues();

//Serial.println(input);

mode = input[2];

dir = input[3];

val = input.substring(4,7).toInt();

//Serial.println(val);

type = input[7];

stateChanged = 1;

}

}

//process commands

switch (mode) {

case '0': //servo , ok

processServo(val, 1);

break;

case '1': //stepper

processStepperMotor(dir, val, 1);

break;

case '2': //dc motor

processDCMotor(dir, val, type);

break;

case '3': //ir sensor -> stepper

processStepperMotor(dir, val, 2);

break;

case '4': //force

break;

case '5': //potentiometer -> servo

processServo(val, 2);

break;

case '6': //switch -> dc motor

processDCMotor(dir, val, '2');

break;

}

}

void sendData(void) {

unsigned long curr\_time = millis();

if (curr\_time - last\_gui\_update\_time > 250) {

//first, read data

int irReading = analogRead(A0);

int irDistance = sharp.distance();

int forceReading = analogRead(A1);

int potReading = analogRead(potpin);

int toggleValue = 1;

if(digitalRead(button\_left)==HIGH) {

toggleValue = 0;

}

String message = "AP";

message = message + " SH " + irReading + " " + irDistance;

message = message + " FO " + forceReading;

message = message + " PO " + potReading;

message = message + " TO " + toggleValue;

message = message + " BT " + pushbutton\_state;

message = message + " EN " + encoderPos;

message = message + " VL " + velocity;

message = message + " OK";

last\_gui\_update\_time = curr\_time;

pushbutton\_state = 0;

}

}

void resetValues(void) {

//reset encoder values so we can keep sending position commands to motor

encoderPos = 0;

last\_encoder\_pos = 0;

//stop dc motor

Motor\_speed\_value = 0;

run\_motor();

i\_error = 0;

i\_error\_vel = 0;

}