Jorge Anton Garcia
Team D – CuBi
Team mates: Laavanye Bahl, Paulo Camasmie, Changsheng Shen (Bobby), Nithin Subbiah Meganathan

ILR01
Feb. 13, 2019

Individual progress

Motor & Sensors Lab

For the motors and sensor lab I was in charge of the micro sensor, the button de-bouncing code, all of the integration, and the communication protocol on the Arduino-side.

The micro sensor consisted of an emitter and receiver which were one centimeter apart and pointed towards each other. When something was placed in between both sensors, the infrared light from the emitter would not hit the receiver and the voltage across receiver would decrease. The datasheet said that the voltage drop across the emitting diode was 1.2V. Since the Arduino provides up to 5V, the voltage drop across the resistor will be approximately 3.8V. The current passing through the circuit should be around 20mA which is below the maximum 50mA the Arduino can supply and in the range specified by the data sheet. Using ohms' law, I calculated that the desired resistance should be about 200 ohms. Regarding the receiving circuit, I used the 1 kohm mentioned as an example in the datasheet. This resulted in very low changes of voltages (in the 0.02V range) when covering and uncovering the receiver and it was almost indistinguishable from noise. I increased the resistance used in the emitter circuit by a factor of 5 and the voltage across the resistor would vary by 0.1V between when there was a detection or not. This change in voltage was now perceivable by the Arduino. To map the voltages to a binary decision of whether there is an obstacle or not, I used a low threshold (0.02V). The results were very reliable when no object was obstructing the IR light path, but would sometimes bounce if there was an obstacle. To reduce this effect, the Arduino predicts that there is not an obstacle if it detects nothing at least three times in a row.

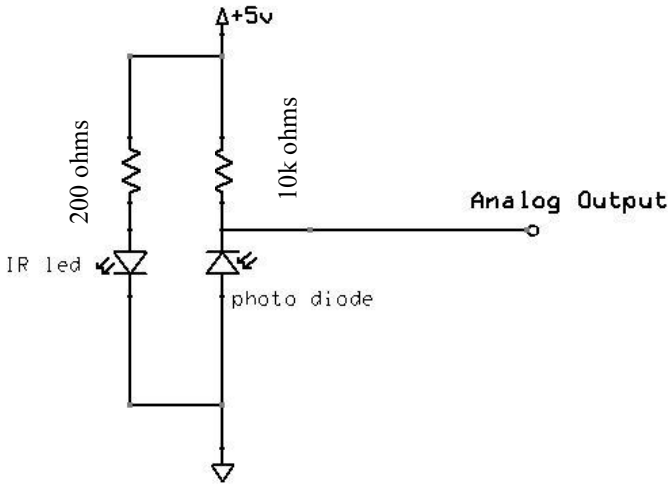

Figure 1. Emitter and receiver circuit for microsensor

To debounce the button, we created an interrupt. It would enter the interrupt at rise and fall. It was only considered a button press if the time interval between the previous accepted button click and the time of the interrupt was over 150ms.

For the integration of all the circuits, I listed all the pins each of the different circuits needed and noted down whether these pins had any specific requirement (ex. an interrupt pin). I then planned out which sensor would interact with each motor and designed where everything would be positioned. I also wired all the circuits onto the breadboard. Regarding the software integration, I decided on how the different components would be interfaced with and created contract requirements for the different functions that interfaced with these components.

There were two modes of operation to control the motors: GUI and sensor based control. In GUI mode, the sensor readings would be sent via serial to a computer and these would be graphed. The GUI could then be used to send desired motor position and speeds. In sensor-based motor control mode, the motors would move depending on sensor values. For example, if the ultrasonic sensor reading values increased, the servo would move clockwise; however, if they decreased, the servo would move counterclockwise. To switch between modes, we used a button. I wrote the functions that would connect most sensor and motor components.

Finally, I worked with Bobby to write the communication protocol. The Arduino would send a list of sensor values separated by commas via serial to the computer. The computer would send three values separated by commas to the Arduino to control each of the motors. The first value was the motor id, the second was a Boolean of whether we wanted to control speed or position, and the last would be either the desired speed or desired angular position. I also had to implement a way to read in Arduino serial which are received as bytes and map them to integers.

CuBi

For CuBi, the three biggest things I focused on was controlling the dynamixels, creating a very granular schedule, and setting team processes in place to work efficiently together. For all of these tasks, I have been working with Nithin.

Nithin and I spent a lot of time getting familiar with the interface to dynamixels and figuring out how to connect and control several of them in series. We worked with Paulo to define mechanical requirements for how they will be used. Some motors need to operate as one and hence need to have a master slave setup, while others need to be connected together, but controlled independently. Since we had received most of these motors from the MRSD inventory, many of the connector components were missing. These were ordered and received, so we can now start mounting all the dynamixels and connecting them together.

We also created a more granular schedule and gant chart that spans the next two months. This involves what tasks correspond to each person and what design decisions need to be made and by who. AAt the same time, I set up the Trello, Slack and Gant chart. Trello will be used for sprint planning and TeamGant will be used for more long-term planning and seeing if we are still on track.

Challenges

Motor & Sensors Lab

The biggest challenge and learning experience in this lab was integration. We had individually divided up who would design the electronics and software of each component and had specified what the function contracts for sensors and motors should look like. This was clearly not enough

and it took me much more than it should have to integrate everything, even after working with teammates to debug specific parts. On the software-side, everyone had managed to run their code using the built-in Arduino loop function, but bugs would appear when functions were called outside of this loop. For example, despite contracts being made, sometimes other auxiliary functions had to be called to update global variables. On the electronics-side, we had not planned which circuit would use which pin. This resulted in having to look back at each individual datasheet to check which components required interrupts or PWM signals. Wires became very messy due to these last minute changes and a lot of time was spent relooking over the specifications. As a team, we realized that this late planning is unacceptable. If it took me more than 20 hours, with team collaboration, to integrate such a simple project, it would take much longer for an entire robot.

Project

The most challenging problem that I have faced on the project-side was how to create a schedule when there are so many uncertainties. We are still deciding on what sensors we will be using and where they will be placed. Designing code for each sensor requires a different set of tasks and timeline. In addition, where we position the sensors also creates a large change in the timeline. For example, if we placed the sensor on top of CuBi's manipulator, tasks need to be created to focus on how we are going to keep the camera as stable as possible on the mechanical-side and how we can account for vibrations on the software-side. If the camera is placed on top of the base, the effort will be placed in creating a tower-like object to put the camera as high as possible to avoid obstruction from the manipulator. To account for this, we have shortened the in-depth timeline and outlined the key decisions that need to be made. Without these design decisions, the team will not advance and continue with further tasks.

Team Work

Nithin: Has worked directly with me to order Dynamixel parts and start designing the software for interfacing with them and creating a schedule. On the lab, he created the functions mapping from voltage value to read-world measurement and designed the circuits for the force-resistance and servo.
Paulo: In charge of the manipulator design. He created a prototype which we all tested and gave him feedback for. After his second iteration, the tray gripper passed our validation test of picking up objects on carpets. For the lab he worked on the PID controller of the DC motor.
Laavanye: Has worked with Bobby to choose the sensors that we will be using for SLAM and grasping. He has started to do object segmentation. In the lab, he designed the stepper motor and ultrasound sensor circuit.
Bobby: Has ordered all the parts for the project. He has also built the turtlebot default configuration and has set it up so it can be controlled using the remote controller. In the lab, he created the GUI, created the communication protocol with me, and soldered several circuits.

Plans

In 2-weeks, the gripper design and placement will be validated, the sensors will be validated and placed, and the gripper mechanism will be completed and 3D printed. We will have our best initial robot design with complete mechanical integration. We should then be able to start working on picking up clutter.

I will work with Nithin to design the wiring for 4 Dynamixels and the code to control it. We will be working with Paulo to ensure that the each Dynamixel has the necessary torque. As always I will also keep track of progress status and make sure we are on track and moving as a team. I will then start working with Laavanye on the vision system.

Task 4 (Sensors and Motor Control Lab) 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?**
     - Typical: -3.6 to 3.6 g
   - **What is the sensor's dynamic range?**
     - BW = 1600 Hz
     - RMS noise = (300 ug) x $\sqrt{(\text{BW x } 1.6)}$ = 0.048
     - Ratio of max signal to noise = $\frac{3.6}{0.048}$ = 75
     - 37.5dB
   - **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?**
     - The goal of the capacitor is to act like a high pass filter. The impedance of a capacitor is 1/jwc. Frequency is proportional to "w" in the impedance equation, so the higher the frequency, the lower the impedance. Therefore, at high frequencies, the capacitor will act like a short allowing them to flow to ground as opposed to the rest of the circuit.
     - It also keeps the charge, to account for small changes in voltage.
   - **Write an equation for the sensor's transfer function.**
     - $(\frac{1g}{300mv} * \frac{1000mv}{1V})$* (V + 1.5) = a
     - 3.3 V + 4.95 = a
   - **What is the largest expected nonlinearity error in g?**
     - 0.3%
   - **How much noise do you expect in the X- and Y-axis sensor signals when the sensor is excited at 25 Hz?**
     - 948ug
   - **How about at 0 Hz? If you can't get this from the datasheet, how would you determine it experimentally?**
     - It cannot be found in the datasheet. I would keep the accelerometer flat and measure the xout and yout voltage values. Then I would map them to acceleration. The readings should be 0g, the difference is error.

2. Signal conditioning
   - Filtering
     - **What problem(s) might you have in applying a moving average?**

        - The moving average is slow to react to quick changes. In the extreme case where you are working with signals which are almost digital, the sudden change in volts from 0 to 5 volts will be modeled as a slow incremental increase. The larger the window, the smoother the signal, but the slower to react to change. In addition, very high peaks of signal, or outliers, can strongly affect the signal for the given time window.
    - **What problem(s) might you have in applying a median filter?**
        - If the median filter is small, it will be able to quickly react to change. However, it will be affected by low frequency noise. For example, if there filter size was four and the values were 1, 2, 80, 90 where the last two are noise, the median filter will return very high numbers for three intervals.
        - The median filter is also more expensive to compute than the mean filter
- 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: 1) which of V1 and V2 will be the input voltage and which the reference voltage; 2) the value of the reference voltage; and 3) the value of Rf/Ri in each case. If the calibration can't be done with this circuit, explain why.**

Gain Offset

: $$V_{out} = V_{in}\left(1 + \frac{R_f}{R_i}\right) - V_{ref}\left(\frac{R_f}{R_i}\right)$$

- The gain is positive because an increase in input voltage results in an increase in output voltage. Since the gain is positive, I used the following equation for both questions below.
- 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 is Vin and V1 is Vref
    - Vref = -3V
    - Rf/Ri = 1
- 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).

- This circuit is not possible as it requires a bias and a gain of 1. To get a gain of 1, Rf/Ri has to be zero, but when this is the case there is no bias, so Vout = Vin.

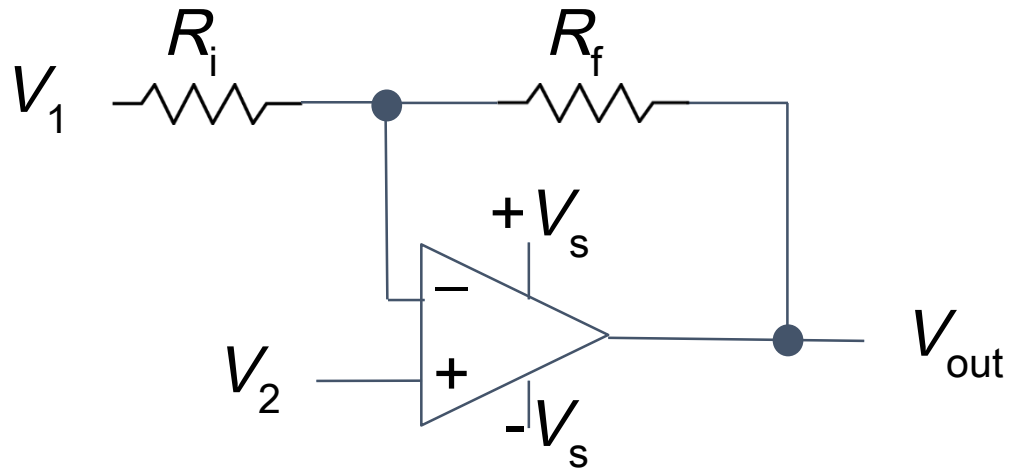

Fig. 1 Opamp gain and offset circuit

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.**
     - The position was calculated by counting the poles passed per unit time. We could then subtract the desired position by the actual position to get the error. The error in position was then multiplied by the proportional term, the sum of the error over time was multiplied by the integral term and the change in error was multiplied times the derivative.
   - **If the system you want to control is sluggish, which PID term(s) will you use and why?**
     - You can increase the proportional and integral terms so that it reacts quicker. The proportional gain will allow the motor to increase its speed when its error is large and the integral term will also cause a quicker reaction to counter act the large initial error.
   - **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?**
     - Increase the integral term because if the error continues to increase over time, it will cause the motor to react to decrease this error.
   - **After applying the control in the previous question, if the system still has overshoot, which PID term(s) will you apply and why?**

- You can increase the derivative control because the moment the signal overshoots, the derivative of the error will be large. For example, if the desired position is 5 and you went from 4.9 to 5.1, then your position error went from -0.1 to 0.1. The change in error is then 0.2 as the errors add up.

```
/*
 *
 */

#include <Servo.h>
// MOTORS:
Servo servo;
int servo_pin = 8;

// SENSORS:
int utlrasound_pin = A0;
int micro_sensor_pin = A1;
int force_input_pin = A2;
const int pwPinUltra = 9;

// BUTTON
int button_pin = 3;
int last_click_time = 0;
int min_time_bet_click = 200;
int STATE = 0;  // GUI MODE is state 0 and Sensor Reaction mode is state 1

// setup interrupt - needed to count every pulse
#define encoder1Pin 2
const int motorOn = 13; // Enables motor driver

//Declare pin functions on Redboard
#define stp 7
#define dir 12
#define MS1 4
#define MS2 5
#define EN  6


float angPos = 0;
float angSpeed = 0.0;


int MAX_NUM_DIGITS = 4;
int sentinel = 44;
int ERROR_VAL = -1;

void setup() {
  // SENSOR SETUP
  pinMode(utlrasound_pin, INPUT);
  pinMode(force_input_pin, INPUT);
  pinMode(pwPinUltra, INPUT);
```

```
  pinMode(micro_sensor_pin, INPUT);

  // BUTTON SETUP
  pinMode(button_pin, INPUT);
  attachInterrupt(digitalPinToInterrupt(button_pin), button_isr, CHANGE);

  // MOTOR SETUP
  servo.attach(servo_pin);
  servo.write(1);

  // initialize digital pins for DC Motor
  pinMode(motorOn, OUTPUT);
  pinMode(encoder1Pin, INPUT_PULLUP);
  // Attach Interrupts for change in pin values encoder
  attachInterrupt(0, stateUpdate, CHANGE);

  // Initialize pins for stepper motor
  pinMode(stp, OUTPUT);
  pinMode(dir, OUTPUT);
  pinMode(MS1, OUTPUT);
  pinMode(MS2, OUTPUT);
  pinMode(EN, OUTPUT);
  resetEDPins(); //Set step, direction, microstep and enable pins to default states

  Serial.begin(9600);
}



void loop() {
  if(STATE == 0) {
      // GUI MODE
      // send all sensor readings
      int no_obstacle = get_microsensor_reading();
      int force = get_force_voltage();
      int pos_ir = get_pos_ultrasound();
      int pos_ultra = read_ultrasound();
      String readings = String(no_obstacle) + "," +  String(force) + "," +  String(pos_ir) + "," +
String(pos_ultra) + "," + String(angPos) + "," + String(angSpeed);
      Serial.println(readings);
      receive_motor_command();

  }
  else if(STATE == 1) {
      move_dc_from_pressure();
      move_stepper_from_ultrasound();
```

```
    move_servo_from_ir();
   }
  else{
   Serial.println("ERROR, state not 0 or 1");
  }
}


int readInt(){
   int sentinel_found = 0;
   int i = 0;

   char input[MAX_NUM_DIGITS];
   int read_int = ERROR_VAL;

   // can only read one byte at a time
   while(!sentinel_found) {
      if (Serial.available() > 0) {
         char c = Serial.read();
         sentinel_found = (c == sentinel);

         if(i>MAX_NUM_DIGITS && !sentinel_found) {
            return read_int;  // will return error
         }

         if(!sentinel_found) {
            input[i++] = c;
         }

      }
   }
   return atoi(input);
}

int prev_stepper_val = 0;
int prev_stepper_ctr = 1;
void receive_motor_command() {
   //update_dc_motor_controls(prev_stepper_ctr, prev_stepper_val);
   updateDCMotor();

    // read all new motor positions
   int id, control, value;
   if(Serial.available()) {
      id = readInt();
    }
   if(Serial.available()) {
```

```
      control = readInt();
    }
   if(Serial.available()) {
      value = readInt();
   }
   else {
    return;
   }

   if(id == 0) {
    if (control == 0){
      moveStepper(value);
    }
   }
   if(id == 1) {
      update_dc_motor_controls(control, value);
      updateDCMotor();
   }
  if(id == 2){
    if(control==0){
      setServoAngle(value);
    }
   }
}

void button_isr(){
   int curr_click_time = millis();
   if(curr_click_time - last_click_time > min_time_bet_click){
      if(digitalRead(button_pin) == HIGH){
        STATE = (STATE == 0) ? 1 : 0;
        //Serial.println("Current State is: " + String(STATE));
        if(STATE == 0) {
            update_dc_motor_controls(1, 0);
            updateDCMotor();
        }
      }
   }
   last_click_time = curr_click_time;
}



void move_servo_from_ir(){
  int no_obstacle = get_microsensor_reading();
  int dist = get_pos_ultrasound();
  int desired_angle = map(dist, 0, 100, 0, 180);
```

```
  if(no_obstacle){
     //servo.write(dist);              // tell servo to go to position in variable 'pos'
     setServoAngle(desired_angle);
  }
}


void move_dc_from_pressure(){
   int force = get_force_voltage();
   // TODO MAP TO 20 to 180 RPM
   int desired_speed = map(force, 20, 10000, 20, 180);
   if(desired_speed < 30) {
     desired_speed = 0;
   }
   update_dc_motor_controls(1, desired_speed);
   updateDCMotor();

}




// 15-100 cm is the range
float get_pos_ultrasound(){
  int pwm_read = analogRead(utlrasound_pin);
  float voltage = pwm_read * 5.0 / 1023.0;
  float pos = -20.31958 + (46500610.31958)/(1 + pow((voltage/7.433096e-8), 0.8057492));   //
TODO: map voltage to distance
  if(pos < 15 || pos > 100){
    pos = -1;
  }
  return pos;
}
```