Sensors and Motors Lab
Individual Lab Report 1
By
Pulkit Goyal

October 14, 2017

Team F - Falcon Eye
Pulkit Goyal
Pratibha Tripathi
Yuchi Wang
Rahul Ramkrishnan
Danendra singh

Individual Progress

For this lab, I was responsible for making the GUI and the state machine for the control of sensors and motor by the user. I also defined the data packet for communication between the arduino and GUI.

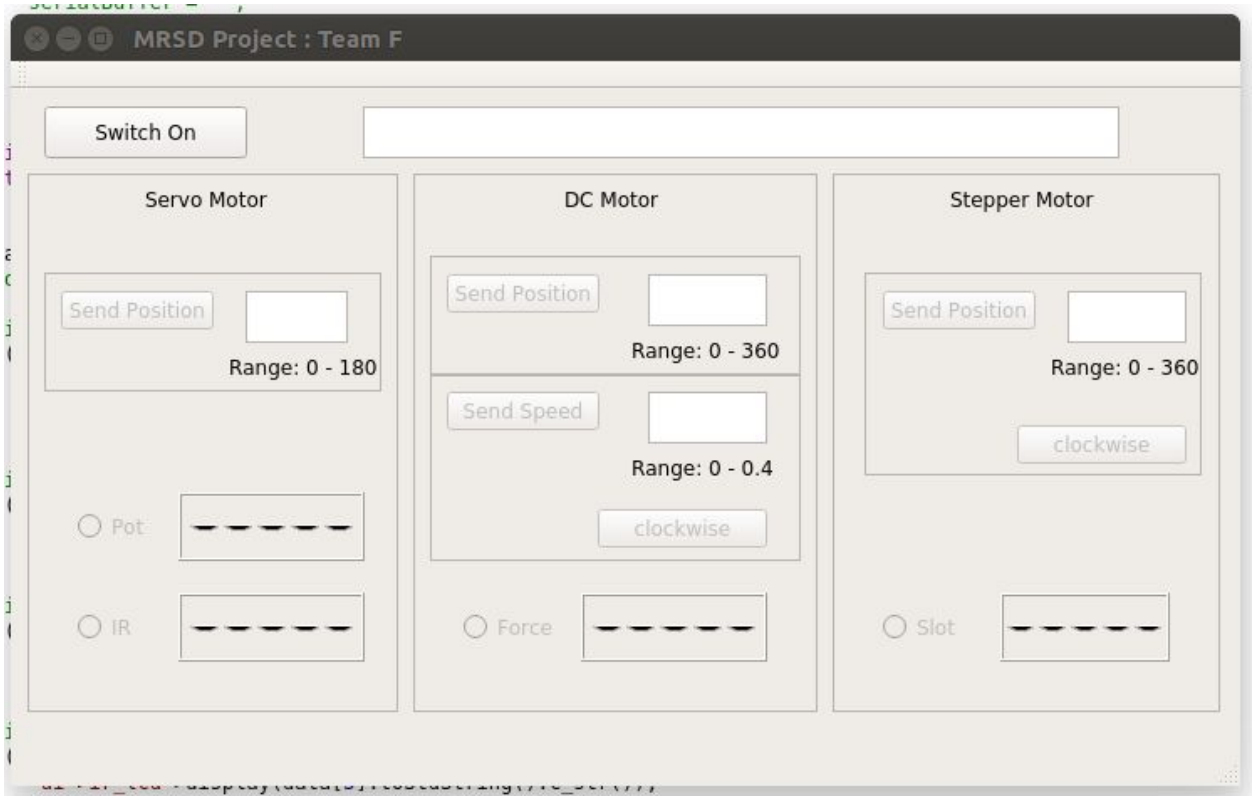
GUI

First step is to identify the existing software for making the GUI which has good support for serial communication. I tried different softwares like QT, processing tool and MATLAB. The MATLAB was quite easy and user friendly whereas processing was too complex in the sense that you need to code in different language. After fair evaluation I finalized on using QT software in ubuntu with the existing QSerial library.

First GUi is initialized to read the serial data from Arduino.

```
// Finding arduino among all the connected components //

foreach(const QSerialPortInfo &serialportinfo, QSerialPortInfo :: availablePorts())
{
    if(serialportinfo.hasProductIdentifier() && serialportinfo.hasVendorIdentifier())
    {
        if((serialportinfo.vendorIdentifier() == arduino_uno_vendor_id )
                && (serialportinfo.productIdentifier() == arduino_uno_product_id))
        {
            arduino_is_available = true;    //arduino is there Yeah!!!
            arduino_uno_port_name = serialportinfo.portName();
        }
    }

}

if(arduino_is_available)
{

    out << "Got the Arduino!!"<<endl;
    arduino->setPortName(arduino_uno_port_name);
    //arduino->open(QSerialPort::ReadWrite);

    if(!arduino->open(QIODevice::ReadWrite))
    {
            out << arduino->error() <<endl ; //error(tr("error %1").arg(arduino->error()));
            out << "Not happening!!\n";
    }
    arduino->setBaudRate(QSerialPort::Baud9600);
    arduino->setDataBits(QSerialPort::Data8);
    arduino->setFlowControl(QSerialPort::NoFlowControl);
    arduino->setParity(QSerialPort::NoParity);
    arduino->setStopBits(QSerialPort::OneStop);
    if(arduino->isOpen())
    {
        out << "I am opened!!\n";
    }
    else
    {
        out << "I am not opened!!\n";
    }

    QObject :: connect(arduino,SIGNAL(readyRead()),this,SLOT(read_serial()));
    out << "All arduino settings done!!"<<endl;

}
else
{
    out << "Couldn't Find the Arduino!!!"<<endl;
}
```

GUI needs to integrate the entire sensor and motor controls hardware and software. A state machine is designed in the GUI to determine and maintain the state of entire system.



The GUI has the following features:

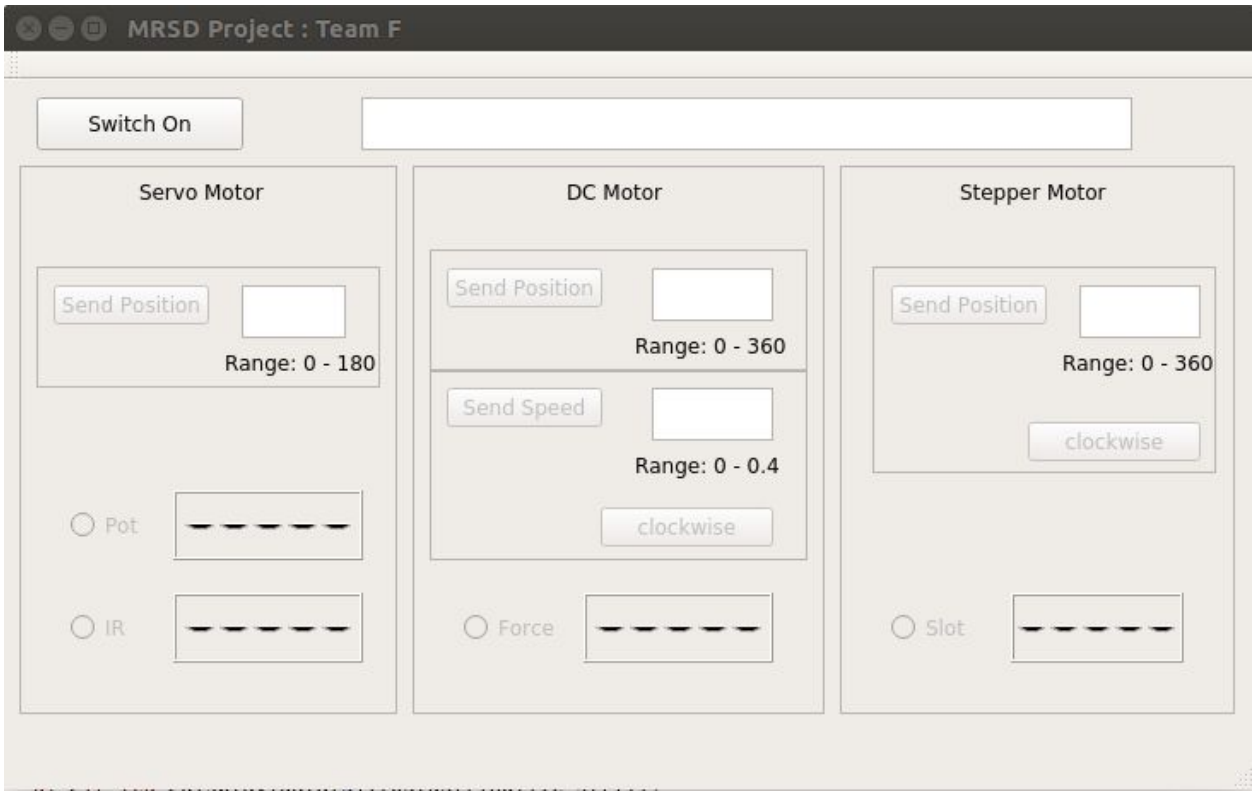
Global State

1. Off/On : When starting the GUI, state of system is ‘Off’ by default. You need to click the button to change the state to Auto ot Manual.
2. Auto: This mode is where different sensors(Pot, IR, Force and Slot) controls the speeds/position/movement of different motors.
3. Manual: This mode requires user to enter the values for different motors for motors to move.

Detailed explanation:

1. **Off**

   User cannot control anything till he/she switches either to auto or manual. All the functionalities are restricted.



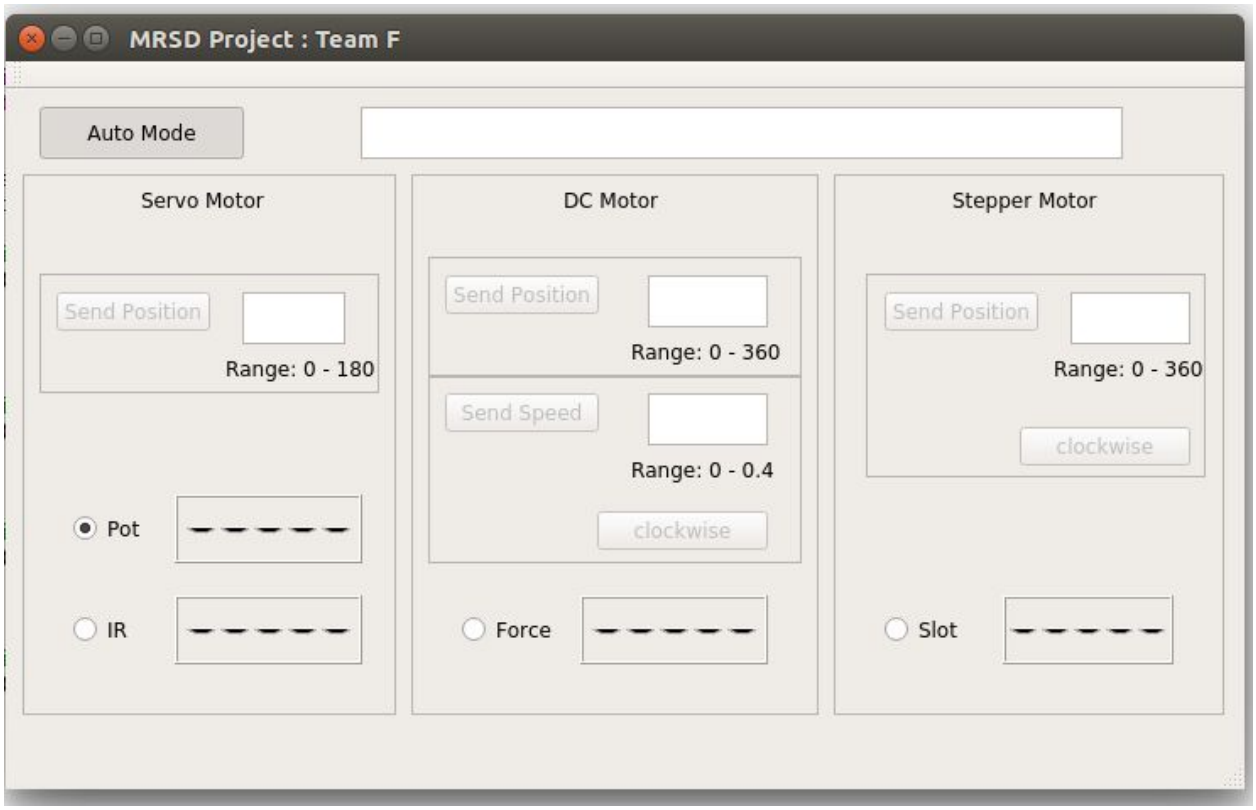
2. **Auto Mode**

   Pot and IR sensor controls the servo motor, Force sensor controls the DC motor, and Slot sensor controls the Stepper motor. You need to select auto mode and the sensor with which you want to control the respective motor.The value of the selected sensor will be displayed on the GUI real-time.

   The text which requires the user to enter the motor values (direction/angle/direction) are commented. User can only select the sensor and see it's real-time values in the GUI.

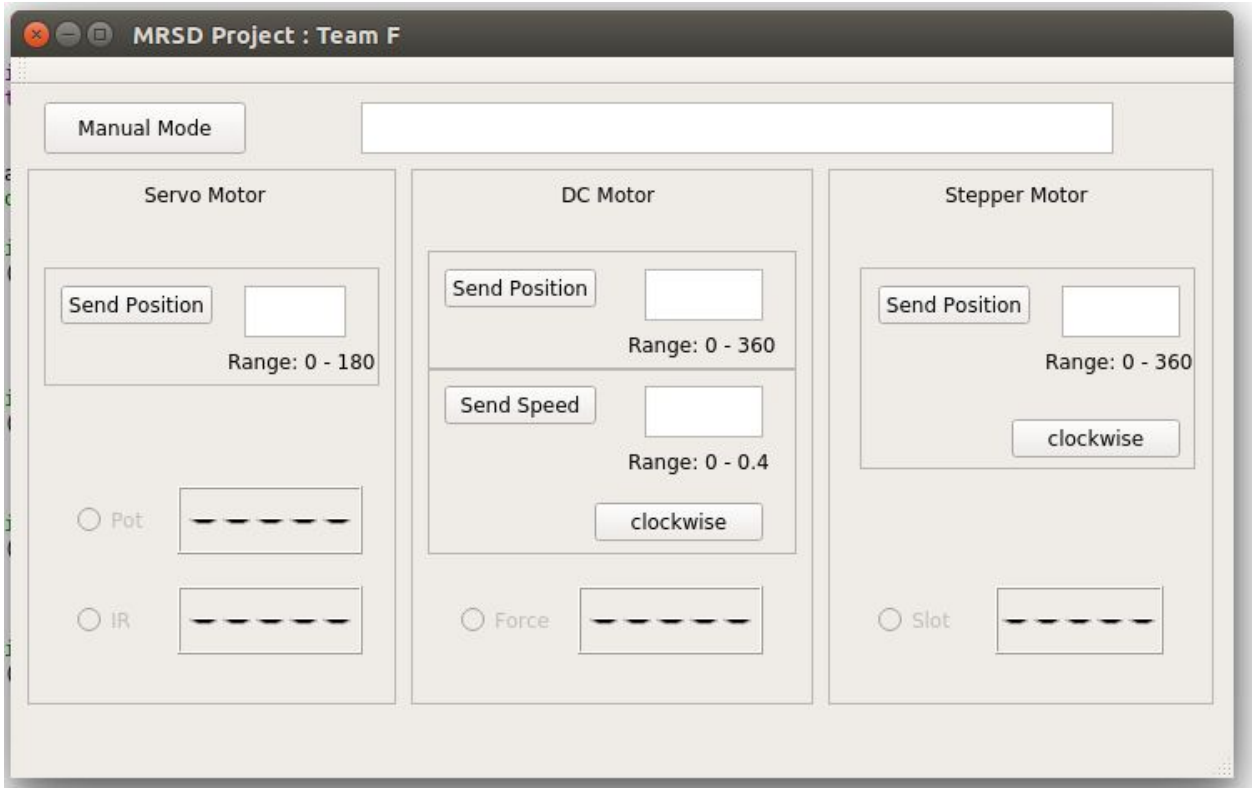
   a. Display different sensor values (Pot, IR, Force and Slot).
   b. Pot: If this sensor is selected, the Servo will move as per the Pot sensor data.

c. IR: If this sensor is selected, the servo will move as per the IR sensor data.
d. Force: If this sensor is selected, the DC motor will move as per the Force sensor data.
e. Slot: If this sensor is selected, the Stepper motor will move as per the Slot sensor data.

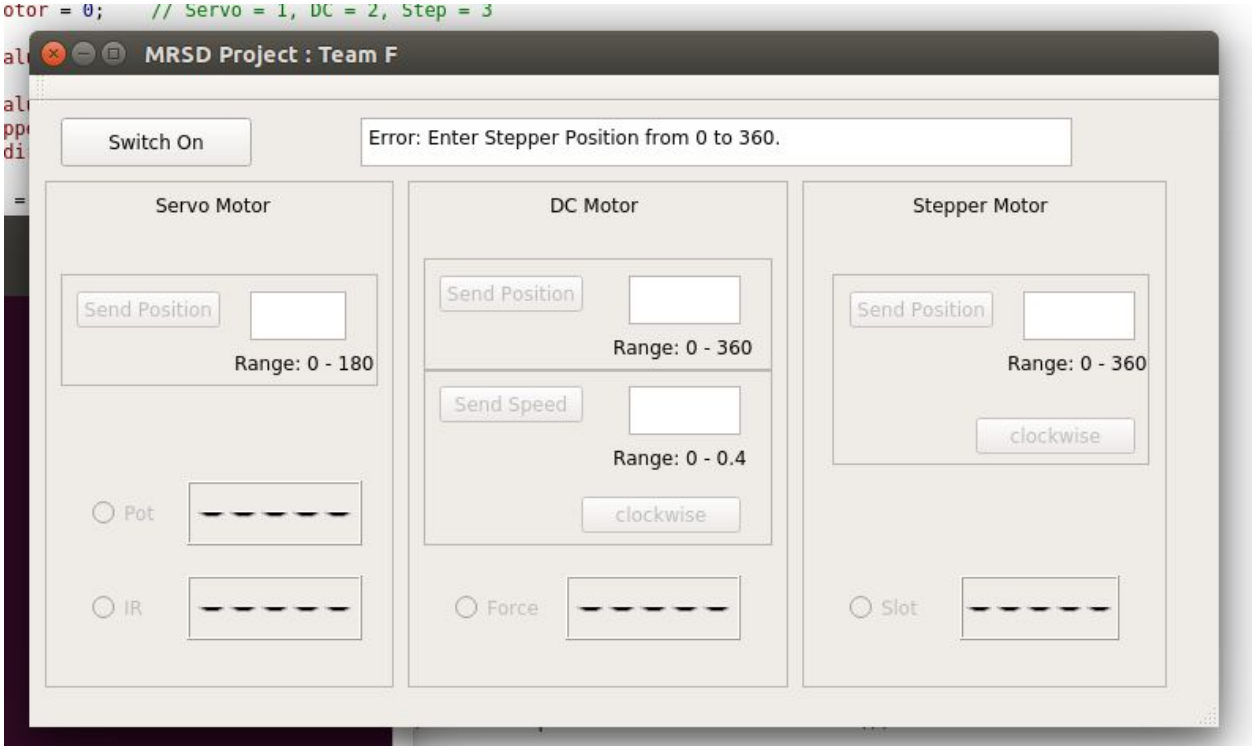


3. **Manual mode** - In this mode, user is supposed to enter the respective motors values to see the motor move.The functionality of selecting the sensor is restricted. User cannot select the sensor to move the respective motor or cannot even see the sensor values in the GUI.
    a. Servo Motor: User is supposed to enter position values for servo motor. Range is supposed to be between 0 and 180. If anything except this range is enter a error is shown in message tab.

b. DC motor
    i. Position: User is supposed to enter the position values for DC motor. Range is supposed to be between 0 and 360. If anything except this range is entered a error message is displayed in message window.
    ii. Speed and Direction: user is supposed to enter the speed in range from 0 to 0.4 and click the button for position, either clockwise or anti-clockwise(clockwise by default). If user enters the value of speed out of the specified the range, an error message is shown in the message window.
c. Stepper Motor: User is supposed to enter position values for servo motor. Range is supposed to be between 0 and 180. If anything except this range is enter a error is shown in message tab.



4. **Message display window**
    a. In case any value is entered in the maunal mode that is out of range. The message is displayed in the message window.
    b. Anything that is supposed to be show to the user is shown in the message window.



Protocol messages for communication between GUI and arduino

Regex for communication

Data from Arduino to GUI (Auto mode)

Type	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
Data	S	Slot Data	Pot Data	Force Data	IR Data	E
Meaning	Starting Byte	Data for slot sensor	Data for pot sensor	Data for force sensor	Data for IR sensor	Ending byte
Range	‘S’	0/1	0-1024	0-256	0-256	‘E’
Example	S	0	102	120	230	E

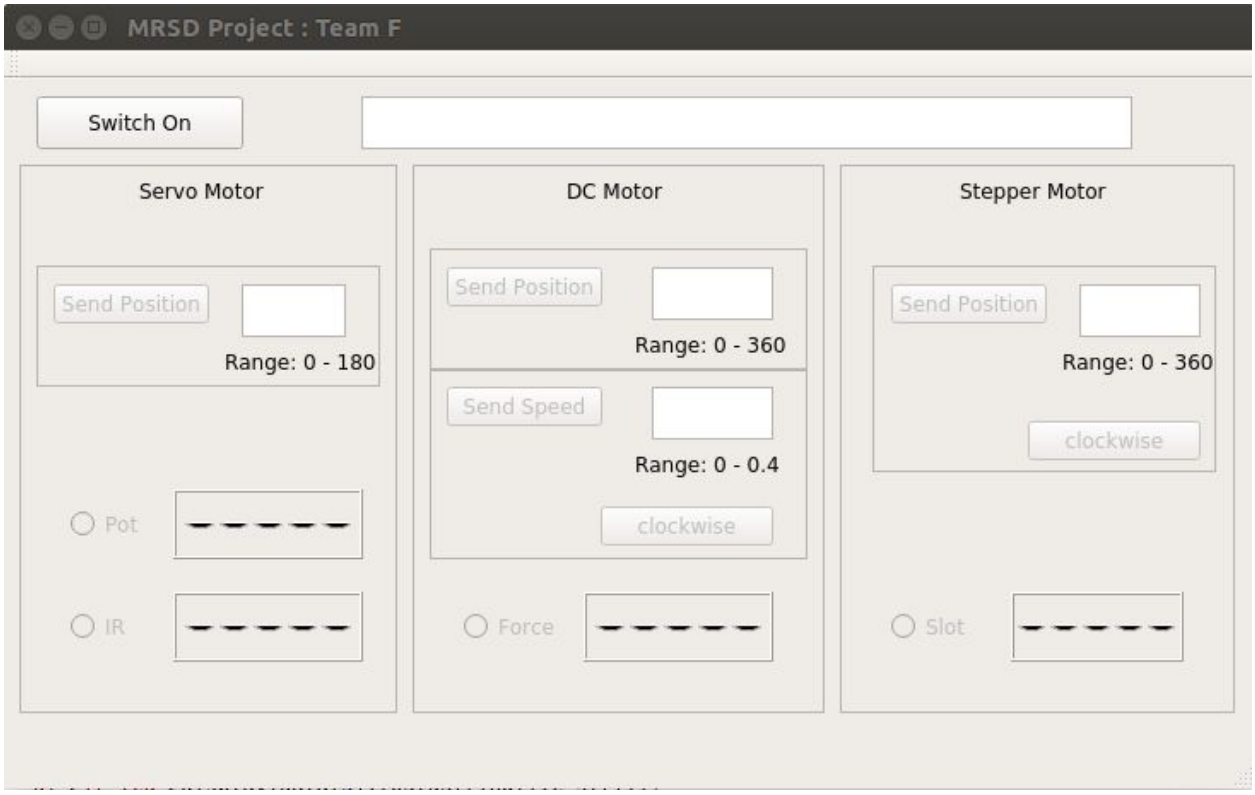
Data from GUI to Arduino (Manual mode)

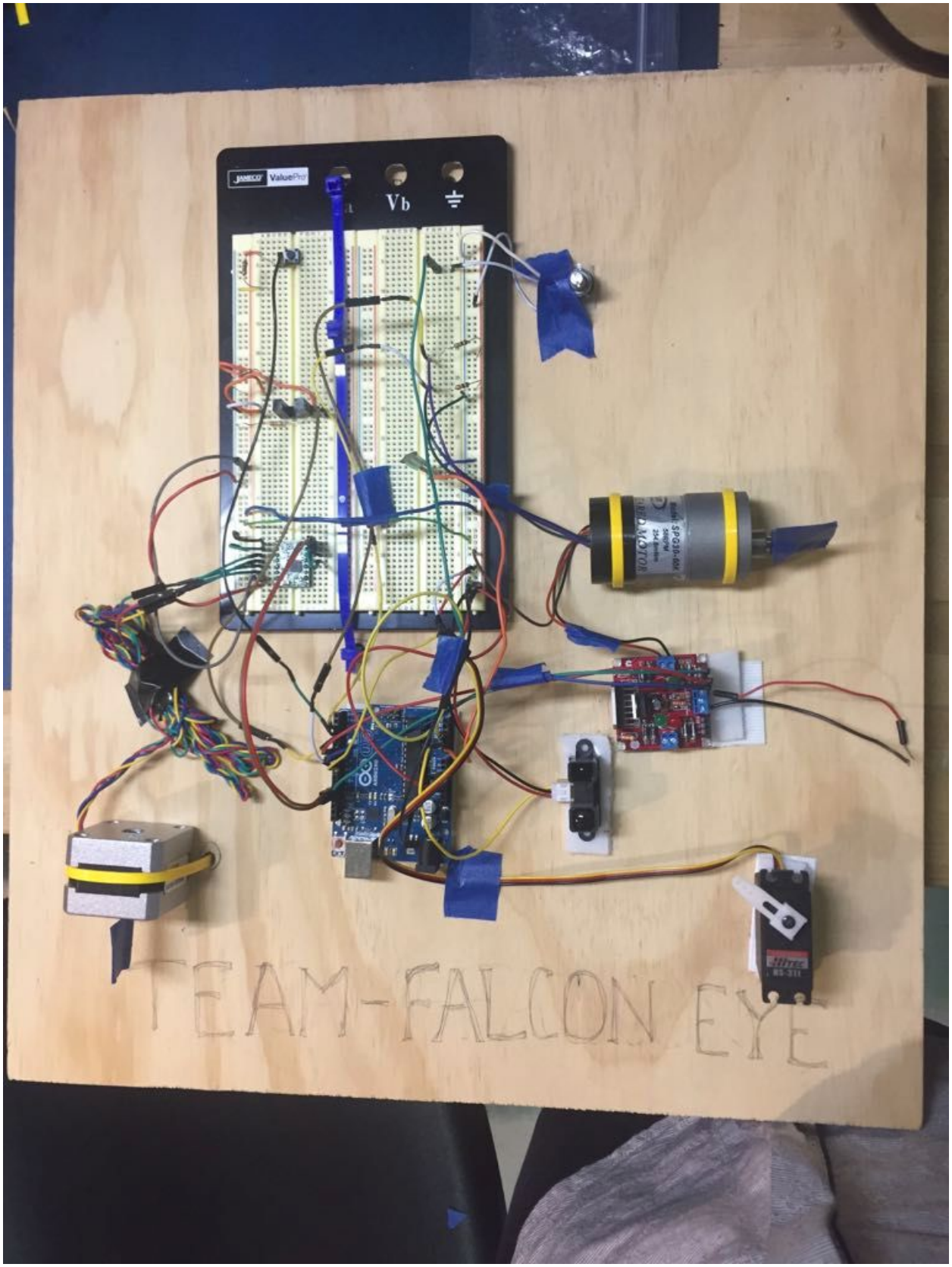
Type	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[]
Data	‘M’	Global_state	state_sensor	State_motor	state_value1	state_value2	‘R’
Meaning	Starting Byte	Auto/Manual	Slot/Pot/Force/IR	Servo/DC_position/DC_speed/Step	Position/Speed	Direction CW/ACW	Ending byte
Range	‘M’	1/2	1/2/3/4	1/2/3/4	0-360/ 0to0.4	1/2	‘R’
Example	SM	1	1	1	230	1	‘R’

Challenges

Serial Monitor of arduino does not work with QT GUI. The usb/serial port can only be opened in single application at one time. It can't be opened simultaneously in multiple applications. Reading data from arduino was not a problem as it can be seen on the QT GUI terminal. But reading the data send by QT GUI on an arduino terminal is not possible. It was quite difficult to debug the issues while sending the data from QT to arduino. I figured a workaround for that, so everytime arduino receives the data from GUI we had to send it back to to GUI so that we can see it on the QT terminal for debugging.

Final System




Vb
TEAM-FALCON EYE

Teamwork

For this task, our team of 5 discussed about the task break down and division among the team members. Once the individual tasks were identified we started to work together to eliminate any miscommunication about the integration in the end. This made the integration at the end of individual modules relatively easy.

This was the first assignment when entire team came together to actually execute a project end to end. We were together from understanding the requirement of the assignment to delivering the final part. We had an assignment deadline due one night before this assignment's deadline(which was in noon next day). Even when the odds were against us, the entire team came together to put an all-nighter in order to complete the assignment. People took complete ownership of the task they were assigned to do. Individual modules including codes and hardware integration was done pretty well, the bugs we encountered during final integration were less than what we expect in a normal project with such a short execution time. It was when the project included hardware as well software.

Yuchi Wang: Force sensor, PID control

In the end Yuchi combined the codes from all the team members, he had pretty tough time pulling of an all-nighter but for sake of project and with constant support from other team members, he supported us like others in team till the last moment.

Pratibha: Servo with IR

Danendra Singh: Stepper and Slot Sensor

Along with their primary role, Pratibha and Danendra played an important role of combining all the sensors and motors on single board so that all the hardware is ready for integration. They worked so meticulously on the hardware side that we didn’t face any issues from hardware side while integrating all the software.

Rahul: Servo motor Potentiometer and DC motor with PID.
Rahul handled a tough part of project, which is PID control of DC motor. We were changing the PID parameters till the last moment of the presentation. He did an awesome job of understanding the PID implementation in the current scope of the project. He also supported the team members in all the other bugs encountered till the very last moment.
It wouldn't have been possible without the 5 people coming together to pull an all-nighter for the execution of the project.

Future Plans

We have two major components in our project: AGV and UAV. We selected Husky as the platform for AGV and Parrot Bebop 2 as the platform for UAV. We ordered the Husky batteries, USB to Serial communication cable and Parrot Beebop Drone 2 so that we can start the work in october. We also arranged a Hololens but after intensively using it we decided to drop the hololens part of our project. We received all the orders in first week of October, which helped us to have more clarity about the future scope of our project. We are focusing in running the husky robot, but we are having a lot of driver issues just to get it started. We are also doing test flights with the drones to understand more about the drone dynamics and control. We still have to figure out the drone control without the RC and with the laptop using a SDK. We are still waiting to acquire a velodyne puck to start with testing out the AGV for SLAM and path planning. We have worked on a backup option of getting a working husky from NREC.

Task 7 (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?
     +/-3.6g(typical) and +/-3g(minimum)
   - What is the sensor's dynamic range?
     20log(0.707 x 3.6/150ug)
   - 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?
     A single 0.1UF capacitor, C_{DC} , is placed close to the ADXL335 supply pins adequately decouples the accelerometer from noise on the power supply.
   - Write an equation for the sensor's transfer function.
     0.3x + 1.5
   - What is the largest expected nonlinearity error in g?
     -/+ 0.3% of full scale
   - How much noise do you expect in the X- and Y-axis sensor signals when the sensor is excited at 25 Hz?
     750 ug/rms
   - How about at 0 Hz? If you can't get this from the datasheet, how would you determine it experimentally?
     Set it to 0 Hz, calculate the deviation and then convert it to rms.

2. Signal conditioning
   - Filtering
     - What problem(s) might you have in applying a moving average?
       The moving average does not filter out high outliers. As it cannot differentiate between two different band of frequencies, it is bad for frequency encoded signals.
     - What problem(s) might you have in applying a median filter?
       Median filter is better than moving average filter as it removes more outliers but it has higher computational cost.

- 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 which of V1 and V2 will be the input voltage and which the reference voltage, the value of the reference voltage, and the value of Rf/Ri in each case. If the calibration can't be done with this circuit, explain why.
    - Your uncalibrated sensor has a range of -1.5 to 1.0V.
    - Your uncalibrated sensor has a range of -2.5 to 2.5V.
      There is no solution. Because it has two equations and three variables.

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.
     First we can give proportional term, i.e difference between current location and desired location. Then to further reduce remaining steady state error apply Integral term i.e sum of previous values. Then manage overshoot apply Derivative term, which can be difference of past two error values.

   - If the system you want to control is sluggish, which PID term(s) will you use and why?
     A high proportional gain term results in high change in output so P term is used for a sluggish 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?
     Integral term(I) is used if system has significant steady state error even after applying the P term. Integral term eliminates remaining steady state error and accelerates the movement of system towards setpoint.

   - After applying the control in the previous question, if the system still has overshoot, which PID term(s) will you apply and why?
     D term, derivative term (anticipatory estimation term) is applied if the system still overshoot. It improves settling time of the system.