Sensor and Motor Lab

Individual Lab Report #1

Pratik Chatrath

Team A – Avengers

Tushar Agrawal, Sean Bryan, Abam Yabroudi, Pratik Chatrath

October 16, 2015

Individual Progress

My role for Task 7 - Sensor Motor Lab was to design and implement the Graphical User Interface for the sensor motor Lab.

Graphical User Interface

I used the QT C++ framework for developing the GUI.

User Interface Design

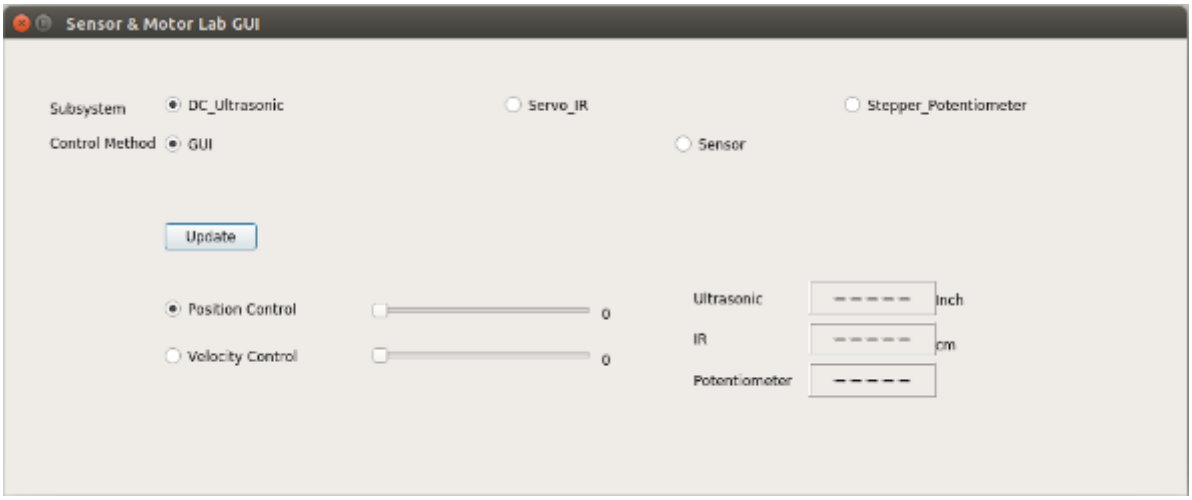

Figure 1

Figure 1 shows our sensor motor lab UI design. Initially I and Tushar designed the way our system will function. We decided to make 3 subsystem

1. DC motor and Ultrasonic Sensor
2. Servo motor and IR sensor
3. Stepper motor and Potentiometer

Each motor is coupled with a sensor. User can select which of the 3 subsystems he/she wants to control. Next the Control Method allows the user to control the subsystem selected using either GUI or Sensor. If the user selects GUI then he/she can choose between Position control or velocity control. On selecting Senor Control Method current sensor reading is reflected on GUI and the motor can be controlled via sensor. Once the user selects all the necessary options and presses “Update” the change reflect on the physical system.

Figure 2 shows the velocity control option selected for DC_Ultrasonic subsystem. On selecting velocity control option the position control slider becomes inactive. Servo and stepper motor have only position control option as shown in figure 3.

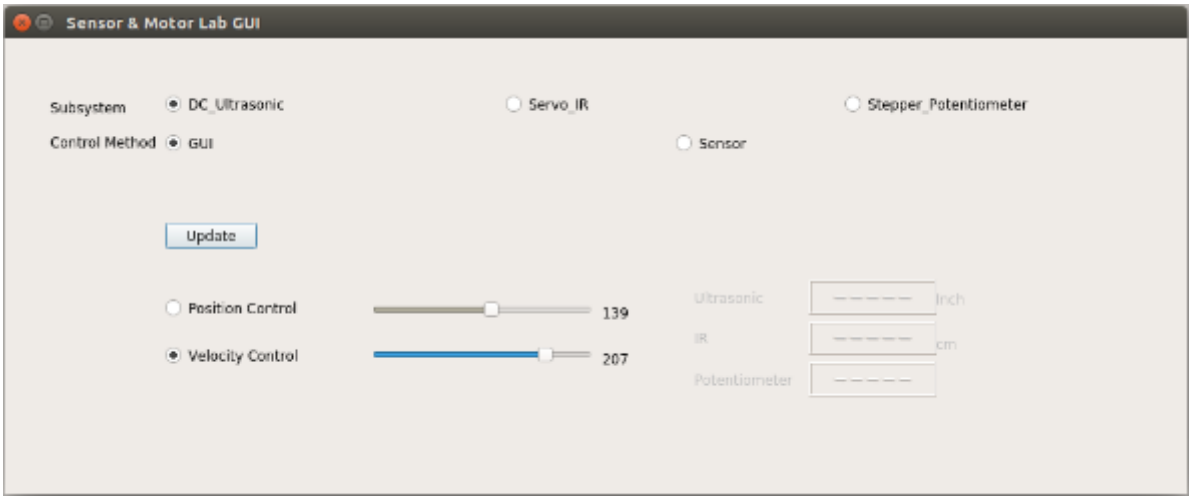

Figure 2

Figure 3

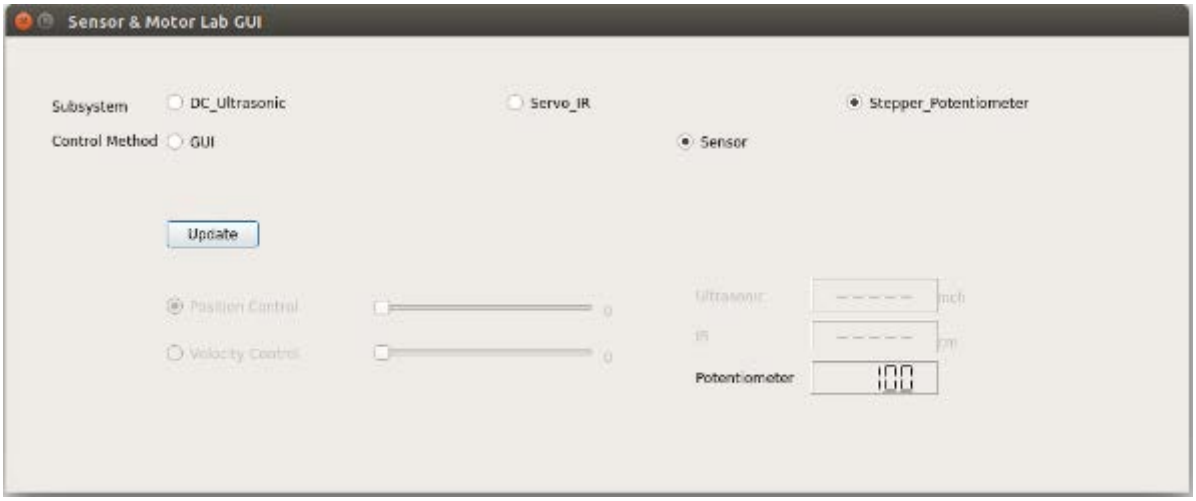

Figure 4

Figure 4 shows the reading of stepper_potentiometer subsystem. The sensor reading are so mapped that they show current position of stepper motor in degrees.

Communication

After designing the User Interface I set up serial communication between QT and Arduino Uno using QTSerialPort library. Once the user selected Sensor Control Method Arduino passed data to GUI. I designed two functions to send and receive data serially.

updateArduino() function passes data from QT to Arduino. For passing data we designed protocol which states that the first digit in every data unit will be "Subsystem", 2nd will be "control method" (GUI or Sensor), 3rd –"position control or velocity control" and last bit will represent "slider value". Each of the data byte will be separated by \n and each bit will be separated by " , " for decoding the signal. Similarly I and Tushar designed **readSerial()** function to send data from Arduino to QT.

Code:

Below is the code for dialog.h header file, dialog.cpp, main.cpp & sensormotorlab.pro.

Dialog.h

```
#ifndef DIALOG_H
#define DIALOG_H

#include <QDialog>
#include <QSerialPort>

namespace Ui {
class Dialog;
}

class Dialog : public QDialog
{
    Q_OBJECT

public:
    explicit Dialog(QWidget *parent = 0);
    ~Dialog();

private slots:
    void on_position_slider_valueChanged(int value);

    void on_velocity_slider_valueChanged(int value);

    void on_radioButton_DCUltrasonic_clicked();

    void on_radioButton_ServoIR_clicked();

    void on_radioButton_StepperPot_clicked();

    void on_radioButton_Gui_clicked();

    void on_radioButton_Sensor_clicked();

    void on_position_control_radio_clicked();

    void on_velocity_control_radio_clicked();

    void updateArduino(QString);

    void readSerial();

    void updateLCD_ultrasonic(const QString);
    void updateLCD_ir(const QString);
    void updateLCD_potentiometer(const QString);

    void on_position_slider_sliderReleased();

    void on_velocity_slider_sliderReleased();

    void on_radioButton_Sensor_pressed();

    void on_update_pushButton_clicked();

private:
    Ui::Dialog *ui;
    QSerialPort *arduino;
    static const quint16 arduino_uno_vendor_id = 9025;
```

Dialog.cpp

```
#include "dialog.h"
#include "ui_dialog.h"
#include <QSerialPort>
#include <QSerialPortInfo>
#include <QDebug>
#include <QtWidgets>

Dialog::Dialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Dialog)
{
    ui->setupUi(this);
    ui->ultrasonic_lcdNumber->display("-----");
    ui->ir_lcdNumber->display("-----");
    ui->potentiometer_lcdNumber->display("-----");

    arduino_is_available = false;
    arduino_port_name = "";

    arduino = new QSerialPort(this);
    serialBuffer = "";
    subsystem = 0;
    input_method = 0;
    control = 0;




    /* code to find product id and vendor id of Arduino Uno
    qDebug() << "Number of available ports: " <<
QSerialPortInfo::availablePorts().length();

    foreach(const QSerialPortInfo &SerialPortInfo,
QSerialPortInfo::availablePorts()){
        qDebug() << "Has vendor ID: " <<
SerialPortInfo.hasVendorIdentifier();
        if(SerialPortInfo.hasVendorIdentifier()){
            qDebug() << "Vendor ID: " << SerialPortInfo.vendorIdentifier();
        }

        qDebug() << "Has Product ID: " <<
SerialPortInfo.hasProductIdentifier();
        if(SerialPortInfo.hasProductIdentifier()){
            qDebug() << "Product ID: " << SerialPortInfo.productIdentifier();
        }

    }
    */

    foreach(const QSerialPortInfo &serialPortInfo,
QSerialPortInfo::availablePorts()){
        if(serialPortInfo.hasVendorIdentifier() &&
serialPortInfo.hasProductIdentifier()){
            if(serialPortInfo.vendorIdentifier() == arduino_uno_vendor_id){
                if(serialPortInfo.productIdentifier() ==
arduino_uno_product_id){
                    arduino_port_name = serialPortInfo.portName();
                     arduino_is_available = true;
```

```
                }
            }
        }
    }

    if(arduino_is_available){
        //open and configure the serialport
        arduino->setPortName(arduino_port_name);
        qDebug() << arduino_port_name;
        if(!arduino->open(QIODevice::ReadWrite)) {
            qDebug() << "Could not open port!" << arduino->error();
         }
        arduino->setBaudRate(QSerialPort::Baud115200);
        arduino->setDataBits(QSerialPort::Data8);
        arduino->setParity(QSerialPort::NoParity);
        arduino->setStopBits(QSerialPort::OneStop);
        arduino->setFlowControl(QSerialPort::NoFlowControl);
        qDebug() << "Hi";
        qDebug() << arduino->isOpen();
        QObject::connect(arduino, SIGNAL(readyRead()), this,
SLOT(readSerial()));
        Dialog::updateArduino(QString("%1").arg(0));


    }else{
        //give error message if not available
        QMessageBox::warning(this, "Port error", "Couldn't find the
Arduino!");
    }

}

Dialog::~Dialog()
{
    if(arduino->isOpen()){
        arduino->close();
    }
    delete ui;
}

void Dialog::on_position_slider_valueChanged(int value)
{
    ui->position_slider_label->setText(QString("%1").arg(value));


    //qDebug() << value;

}

void Dialog::on_velocity_slider_valueChanged(int value)
{
    ui->velocity_slider_label->setText(QString("%1").arg(value));




}

void Dialog::on_radioButton_DCUltrasonic_clicked()
{
    subsystem = 0;
    if(ui->radioButton_Gui->isChecked()){
```

```
        ui->position_slider_label->setEnabled(true);
        ui->velocity_slider_label->setEnabled(true);
        ui->position_slider->setEnabled(true);
        ui->velocity_slider->setEnabled(true);
        ui->position_control_radio->setEnabled(true);
        ui->velocity_control_radio->setEnabled(true);
        ui->potentiometer_lcdNumber->setDisabled(true);
        ui->potentiometer_label->setDisabled(true);
        ui->ultrasonic_lcdNumber->setDisabled(true);
        ui->ultrasonic_label->setDisabled(true);
        ui->ir_lcdNumber->setDisabled(true);
        ui->IR_label->setDisabled(true);
        ui->CM_label->setDisabled(true);
        ui->Inch_label->setDisabled(true);
    }
    else if(ui->radioButton_Sensor->isChecked()){
        ui->position_slider_label->setDisabled(true);
        ui->velocity_slider_label->setDisabled(true);
        ui->position_slider->setDisabled(true);
        ui->velocity_slider->setDisabled(true);
        ui->position_control_radio->setDisabled(true);
        ui->velocity_control_radio->setDisabled(true);
        ui->potentiometer_lcdNumber->setDisabled(true);
        ui->potentiometer_label->setDisabled(true);
        ui->ultrasonic_lcdNumber->setEnabled(true);
        ui->ultrasonic_label->setEnabled(true);
        ui->ir_lcdNumber->setDisabled(true);
        ui->IR_label->setDisabled(true);
        ui->CM_label->setDisabled(true);
        ui->Inch_label->setEnabled(true);
    }



}

void Dialog::on_radioButton_ServoIR_clicked()
{
    subsystem = 1;

    if(ui->radioButton_Gui->isChecked()){
        ui->position_slider_label->setEnabled(true);
        ui->velocity_slider_label->setDisabled(true);
        ui->position_slider->setEnabled(true);
        ui->velocity_slider->setDisabled(true);
        ui->position_control_radio->setEnabled(true);
        ui->velocity_control_radio->setDisabled(true);
        ui->potentiometer_lcdNumber->setDisabled(true);
        ui->potentiometer_label->setDisabled(true);
        ui->ultrasonic_lcdNumber->setDisabled(true);
        ui->ultrasonic_label->setDisabled(true);
        ui->ir_lcdNumber->setDisabled(true);
        ui->IR_label->setDisabled(true);
        ui->CM_label->setDisabled(true);
        ui->Inch_label->setDisabled(true);
    }
    else if(ui->radioButton_Sensor->isChecked()){
        ui->position_slider_label->setDisabled(true);
        ui->velocity_slider_label->setDisabled(true);
        ui->position_slider->setDisabled(true);
        ui->velocity_slider->setDisabled(true);
        ui->position_control_radio->setDisabled(true);
```

```
        ui->velocity_control_radio->setDisabled(true);
        ui->potentiometer_lcdNumber->setDisabled(true);
        ui->potentiometer_label->setDisabled(true);
        ui->ultrasonic_lcdNumber->setDisabled(true);
        ui->ultrasonic_label->setDisabled(true);
        ui->ir_lcdNumber->setEnabled(true);
        ui->IR_label->setEnabled(true);
        ui->CM_label->setEnabled(true);
        ui->Inch_label->setDisabled(true);
    }
}

void Dialog::on_radioButton_StepperPot_clicked()
{
    subsystem = 2;

    if(ui->radioButton_Gui->isChecked()){
        ui->position_slider_label->setEnabled(true);
        ui->velocity_slider_label->setDisabled(true);
        ui->position_slider->setDisabled(true);
        ui->velocity_slider->setDisabled(true);
        ui->position_control_radio->setEnabled(true);
        ui->velocity_control_radio->setDisabled(true);
        ui->potentiometer_lcdNumber->setDisabled(true);
        ui->potentiometer_label->setDisabled(true);
        ui->ultrasonic_lcdNumber->setDisabled(true);
        ui->ultrasonic_label->setDisabled(true);
        ui->ir_lcdNumber->setDisabled(true);
        ui->IR_label->setDisabled(true);
        ui->CM_label->setDisabled(true);
        ui->Inch_label->setDisabled(true);
    }
    else if(ui->radioButton_Sensor->isChecked()){
        ui->position_slider_label->setDisabled(true);
        ui->velocity_slider_label->setDisabled(true);
        ui->position_slider->setDisabled(true);
        ui->velocity_slider->setDisabled(true);
        ui->position_control_radio->setDisabled(true);
        ui->velocity_control_radio->setDisabled(true);
        ui->potentiometer_lcdNumber->setEnabled(true);
        ui->potentiometer_label->setEnabled(true);
        ui->ultrasonic_lcdNumber->setDisabled(true);
        ui->ultrasonic_label->setDisabled(true);
        ui->ir_lcdNumber->setDisabled(true);
        ui->IR_label->setDisabled(true);
        ui->CM_label->setDisabled(true);
        ui->Inch_label->setDisabled(true);
    }
}

void Dialog::on_radioButton_Gui_clicked()
{
    input_method = 0;
    ui->CM_label->setDisabled(true);
    ui->Inch_label->setDisabled(true);
    ui->position_slider->setEnabled(true);
    //ui->velocity_slider->setEnabled(true);
    ui->position_control_radio->setEnabled(true);
    //ui->velocity_control_radio->setEnabled(true);
    ui->potentiometer_lcdNumber->setDisabled(true);
    ui->potentiometer_label->setDisabled(true);
    ui->ultrasonic_lcdNumber->setDisabled(true);
```

```
    ui->ultrasonic_label->setDisabled(true);
    ui->ir_lcdNumber->setDisabled(true);
    ui->IR_label->setDisabled(true);
    if(ui->radioButton_DCUltrasonic->isChecked()){
        ui->velocity_control_radio->setEnabled(true);
        ui->velocity_slider->setEnabled(true);
        ui->position_control_radio->setEnabled(true);
        ui->position_slider->setEnabled(true);
    }
    else if(ui->radioButton_ServoIR->isChecked()){
        ui->velocity_control_radio->setDisabled(true);
        ui->velocity_slider->setDisabled(true);
        ui->position_control_radio->setEnabled(true);
        ui->position_slider->setEnabled(true);

    }
    else if(ui->radioButton_StepperPot->isChecked()){
        ui->velocity_control_radio->setDisabled(true);
        ui->velocity_slider->setDisabled(true);
        ui->position_control_radio->setEnabled(true);
        ui->position_slider->setEnabled(true);

    }

}

void Dialog::on_radioButton_Sensor_clicked()
{

}

void Dialog::on_position_control_radio_clicked()
{
    control = 0;
    ui->position_slider->setEnabled(true);
    ui->velocity_slider->setDisabled(true);

}

void Dialog::on_velocity_control_radio_clicked()
{
    control = 1;
    ui->position_slider->setDisabled(true);
    ui->velocity_slider->setEnabled(true);

}

void Dialog::updateArduino(QString command)
{
    if(arduino->isWritable()){
        QString str;
        str.append(QString("%1,").arg(subsystem));
        str.append(QString("%1,").arg(input_method));
        str.append(QString("%1,").arg(control));
        //qDebug() << str;

        str.append(QString("%1\n").arg(command));
        qDebug() << str;

        arduino->write(str.toStdString().c_str());
    }else{
        qDebug() << "Couldn't write to serial";
```

```
    }
}

void Dialog::readSerial()
{

    serialData = arduino->readAll();
    serialBuffer += QString::fromStdString(serialData.toStdString());
    QStringList bufferSplit = serialBuffer.split("\r\n");
    if(bufferSplit.length() < 2){
        serialData = arduino->readAll();
         serialBuffer += QString::fromStdString(serialData.toStdString());

     }else{
         //bufferSplit(1) is a good value
         qDebug() << bufferSplit;
         QString tmp = bufferSplit[0];
         QStringList DataReceived = tmp.split(",");
         subsystem_value = DataReceived[0];
         sen_val = DataReceived[1];
         int valtmp;
         if(QString::compare(subsystem_value, "0") == 0 ||
QString::compare(subsystem_value, "1") == 0){
             valtmp = sen_val.toInt()/10;
         }
         else if(QString::compare(subsystem_value, "2") == 0){
             valtmp = ((sen_val.toInt()*360)/1023);

         }
         sen_val = QString::number(valtmp);
         qDebug() << sen_val << "subsystem" << subsystem ;
         //Dialog::updateLCD(bufferSplit[1]);
         serialBuffer = "";
         if(QString::compare(subsystem_value, "0") == 0){
             ui->ultrasonic_lcdNumber->setEnabled(true);
             ui->ultrasonic_label->setEnabled(true);
             ui->ir_lcdNumber->setDisabled(true);
             ui->IR_label->setDisabled(true);
             ui->potentiometer_lcdNumber->setDisabled(true);
             ui->potentiometer_label->setDisabled(true);
             //QString updated_sen_val
             //Dialog::updateLCD(QString("%1").arg(sen_val));
             Dialog::updateLCD_ultrasonic(sen_val);
             //Debug() << "In 0";

         }
         else if(QString::compare(subsystem_value, "1") == 0){
             ui->ir_lcdNumber->setEnabled(true);
             ui->IR_label->setEnabled(true);
             ui->potentiometer_lcdNumber->setDisabled(true);
             ui->potentiometer_label->setDisabled(true);
             ui->ultrasonic_lcdNumber->setDisabled(true);
             ui->ultrasonic_label->setDisabled(true);
              Dialog::updateLCD_ir(sen_val);
         }
         else if(QString::compare(subsystem_value, "2") == 0){
             ui->potentiometer_lcdNumber->setEnabled(true);
             ui->potentiometer_label->setEnabled(true);
             ui->ultrasonic_lcdNumber->setDisabled(true);
             ui->ultrasonic_label->setDisabled(true);
             ui->ir_lcdNumber->setDisabled(true);
             ui->IR_label->setDisabled(true);
```

```
                Dialog::updateLCD_potentiometer(sen_val);

            }

        }

    }

void Dialog::updateLCD_ultrasonic(const QString sensor_reading)
{
    ui->ultrasonic_lcdNumber->display(sensor_reading);
}


void Dialog::updateLCD_ir(const QString sensor_reading)
{
    ui->ir_lcdNumber->display(sensor_reading);
}


void Dialog::updateLCD_potentiometer(const QString sensor_reading)
{
    ui->potentiometer_lcdNumber->display(sensor_reading);
}

void Dialog::on_position_slider_sliderReleased()
{
    int value = ui->position_slider->sliderPosition();
    Dialog::updateArduino(QString("%1").arg(value));
    //qDebug() << value;
}

void Dialog::on_velocity_slider_sliderReleased()
{
    int value = ui->velocity_slider->sliderPosition();
    Dialog::updateArduino(QString("%1").arg(value));

}

void Dialog::on_radioButton_Sensor_pressed()
{
    input_method = 1;
    qDebug() << "I am here";
    ui->position_slider->setDisabled(true);
    ui->velocity_slider->setDisabled(true);
    ui->position_control_radio->setDisabled(true);
    ui->velocity_control_radio->setDisabled(true);
    ui->potentiometer_lcdNumber->setEnabled(true);
    ui->potentiometer_label->setEnabled(true);
    ui->ultrasonic_lcdNumber->setEnabled(true);
    ui->ultrasonic_label->setEnabled(true);
    ui->ir_lcdNumber->setEnabled(true);
    ui->IR_label->setEnabled(true);
    ui->CM_label->setEnabled(true);
    ui->Inch_label->setEnabled(true);
    ui->position_slider_label->setDisabled(true);
    ui->velocity_slider_label->setDisabled(true);

}

void Dialog::on_update_pushButton_clicked()
{
```

```
    Dialog::updateArduino(QString("%1").arg(0));
}
```

Main.cpp

```
#include "dialog.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Dialog w;
    w.show();
    w.setWindowTitle("Sensor & Motor Lab GUI");

    return a.exec();
}
```

Sensormotorlab.pro

```
#-------------------------------------------------
#
# Project created by QtCreator 2015-10-13T17:18:53
#
#-------------------------------------------------

QT       += core gui serialport

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = sensormotorlab
TEMPLATE = app


SOURCES += main.cpp\
        dialog.cpp

HEADERS  += dialog.h

FORMS    += dialog.ui
```

Challenges

I faced the following 3 challenges while designing GUI for the Lab.

1. Downloading and installing QT took a while. Windows doesn't have a default C++ compiler. On Linux I faced issue getting the serial port to work. Finally with help from Tushar I figured out that I had to run QT as root in Ubuntu to get the serial port work.
2. I had never worked before with QT and C++ so I took me time to understand QT framework.
3. Parsing serial data with no errors was a challenge.

Teamwork

Our team divided the lab's task as follows. Tushar worked with DC motor and ultrasonic sensor. He also integrated the whole Arduino code and help me with QT. I was responsible for designing GUI. Adam interfaced Servo motor with IR sensor and integrating the whole electrical system. Sean controlled Stepper motor using potentiometer and built the mechanical system required for the task.

Plans for coming week

By next weekend I plan to decide obstacle avoidance sensor system for our UAV. I will test combination of Ultrasonic and IR sensor to decide number and position of the sensor combination on the model of our UAV that Sean will build. I will check the detection cone of each sensor, angles and distance when sensors start to interface with each other. I will test the sensor system in different settings like outdoor environment, varying lighting conditions and near a moving motor to see if that induces noise in the sensor readings.