SENSORS AND MOTORS LAB

Astha Prasad

Team F / ADD_IN

Teammates: Daniel Berman, Nikhil Baheti, Ihsane Debbache

ILR #1

October 16th, 2015

Individual Progress

For the Sensors and Motors lab, I was responsible for implementing the operation of the Servo Motor in accordance with Proximity Sensor readings, as well as the pushbutton debouncing.

Approach

For this assignment, Team F decided to work on the RAMBo board which houses an ATmega2560 microcontroller. This is the board that our Makergear M2 3D printer uses and hence, we thought getting acquainted with the board would be a useful exercise. Since we only had 2 of these boards, initial testing of the Servo motor and Proximity sensor was done on the Arduino Uno board. Once tested, the hardware and software was integrated with that of the team.

Implementation

The proximity sensor was a SHARP IR 2Y0A21 sensor that has a distance measuring range of 10 to 80 cm. This was interfaced with the servo motor which is not modifiable to continuous rotation, i.e., it is limited to move between 0 to 180 degrees. They are shown in the figures below:

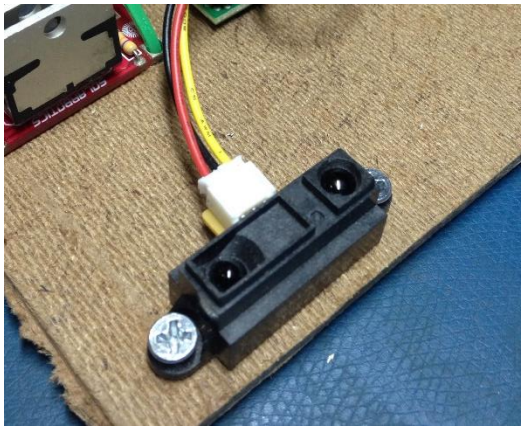

Figure 1 : Sharp IR 2Y0A21 Proximity Sensor

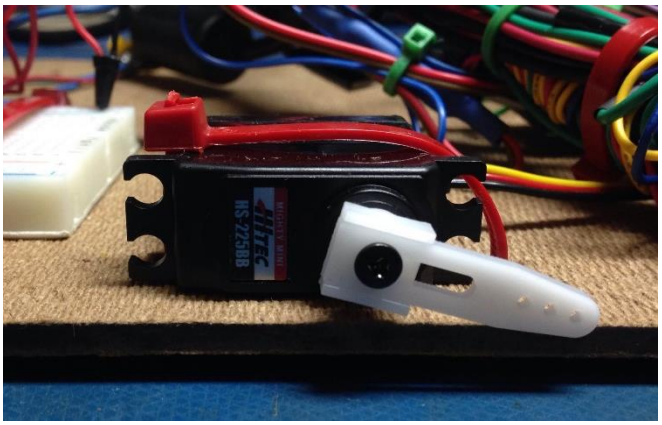

Figure 2 : Servo Motor

There exists an Arduino library for SharpIR sensors that proved to be very useful for this exercise. An object called ‘sharp’ is initialized and fed the following data:

- Ir: Pin number for analog input (where sensor is attached).
  - Pin used: 62
- Avg: Number of readings over which average will be calculated
  - 20
- Tolerance: How similar the value has to be from the last value to be taken as valid. This is a percentage value.
  - 70 i.e., A reading must be at least 70% of the previous reading to be considered as valid.
- Sensor Range: The range of the sensor in cm.
  - 1080 i.e. ,The sensor can measure distance within a range of 10cm to 80cm

Proximity sensor readings are read through the analog input pin and then constrained between 7 and 50. These readings are then mapped to a value between 0 and 180. This value was then multiplied by a certain gain that the user can enter through the GUI. This mapped and scaled value is then fed to the servo motor connected to a PWM pin of the board. Hence, with a gain of 1, the servo motor moves from 0 to 180 degrees as an object goes from 7 to 50 cm form the proximity sensor. As the gain increases, the servo becomes more sensitive to movement.

A push button was connected to the board and debounced. Debouncing was performed using the following algorithm:

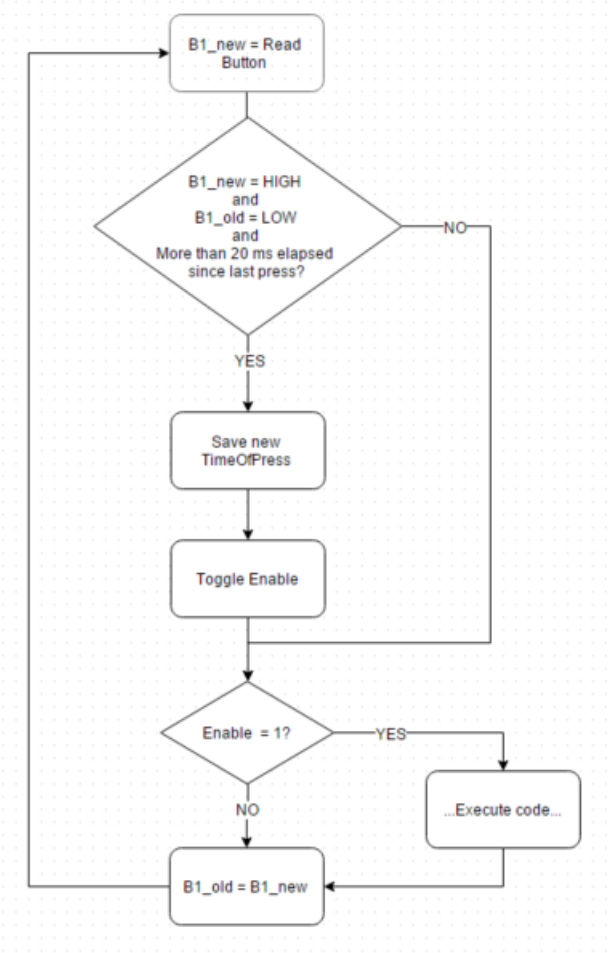

Figure 3 : Flowchart for control loop of Servo Motor

Every time the push button was pressed, a Boolean variable 'Enable' was toggled. This variable dictated whether or not the sensor reading was to be written to the stepper motor. In addition to this button enable, the GUI also permitted the user to enable/disable stepper motor.

Challenges

A challenge we had foreseen was software integration. Since my code was part of a much larger program, it was slightly unclear what functions I needed to write in order for the GUI to interface smoothly. This was easily overcome though as we drew up a framework of the main program and decided on the flow of the program. This gave all of us a clearer picture of what our individual code was to look like. Dan, who was in charge of the GUI, made it clear what variable names we were to work with which greatly helped maintain a standard across all our codes.

A major take away from overcoming this challenge was the importance of a coding standard for a team coding project. We have now laid groundwork for future exercises such as this one and I think they will help save a lot of time and effort during integration.

Teamwork

The team split up the tasks in the following manner:

Daniel Berman

Dan was in charge of the GUI which was developed on QT. He played a major role in defining the standards for naming and coding that we all used.

Nikhil Baheti

Nikhil was in charge of the DC motor control and the force sensor. He implemented the PID controller with the help of Dan.

Ihsane Debbache

Ihsane programmed the stepper motor and the potentiometer and drew up the initial main loop of the program.

Future Goals

For the team project, the plan for this week is to be able to pause at pre-decided layers during the printing of a 3D object. This entails sending G-Code commands to move the extruder to a home position whenever it encounters a layer that has been designated to be an insertion layer by the user.

Code for Servo motor and Proximity sensor

Servo.h:

```
#ifndef servo

#define servo

#define model 1080

void setupServo();

void controlServo();

void readProx();

#endif
```

Servo:

```
#include "pindefinitions.h"

#include "protocoldefinitions.h"

#include "servo.h"

#include "main.h"


#include <Servo.h>

#include <SharpIR.h>


SharpIR sharp(SharpPin, 20, 50, model);  //(ReadPin, #readings library will take before calculating mean distance, #min difference between 2 consecutive measurements to be valid, range of sensor)

Servo MyServo;

bool B1_old = LOW;

bool B1_debounced = LOW;

bool B1_new = LOW;

bool buttonEnable = LOW;

unsigned long timeOfLastButton1 = 0;

int debounceInterval = 20;



extern stateVariables sv;
```

```
void setupServo(){

  MyServo.attach(ServoPin);
  pinMode(ServoPin, OUTPUT);
  pinMode (SharpPin, INPUT);
  pinMode(Button1, INPUT);
  pinMode(ButtonLED, OUTPUT);
}

void controlServo()
{

  B1_new = digitalRead(Button1);
  if(B1_new == HIGH && B1_old == LOW && millis() - timeOfLastButton1 > debounceInterval){
      timeOfLastButton1 = millis();
      buttonEnable = !buttonEnable;
      sv.buttonStatus = buttonEnable;
  }

  if (sv.servoEnabled && buttonEnable){
    sv.servoCurrentPos = map(sv.servoSensorVal, 7,50, 0,180*sv.servoGain);
    sv.servoCurrentPos = constrain(sv.servoCurrentPos,0,180);
    MyServo.write(sv.servoCurrentPos);
  }

  B1_old = B1_new;

}

void enableServo(bool enable){
  sv.servoEnabled = enable;
}
```

```
void readProx(){
   sv.servoSensorVal = sharp.distance();
   sv.servoSensorVal = constrain(sv.servoSensorVal,7,50);
}
```