Sensors and Motors Lab

Nikhil Baheti

Team F: ADD_IN

Teammates: Nikhil Baheti, Dan Berman and Astha Prasad

ILR01

October 16th, 2015

1 Individual Progress

For the sensors and motors assignment, I was responsible for implementing the DC geared motor and the force sensor. Also, as a part of the ADD_IN project, I was responsible to design and make a 3D model of the coupler that can help convert the nozzle extrusion plane to a plane that is at an angle of 45 degrees to it.

1.1 Sensors and Motors Lab

In the sensors and motors lab I was responsible to implement the DC motor and the force sensor. To start the implementation I first looked into the RAMBo board interfacing pins. Once the interface pins were identified, the implementation of the modules was done which is explained in the following section. Figure 1.1 shows the abstract algorithm and highlights the part I have worked on.

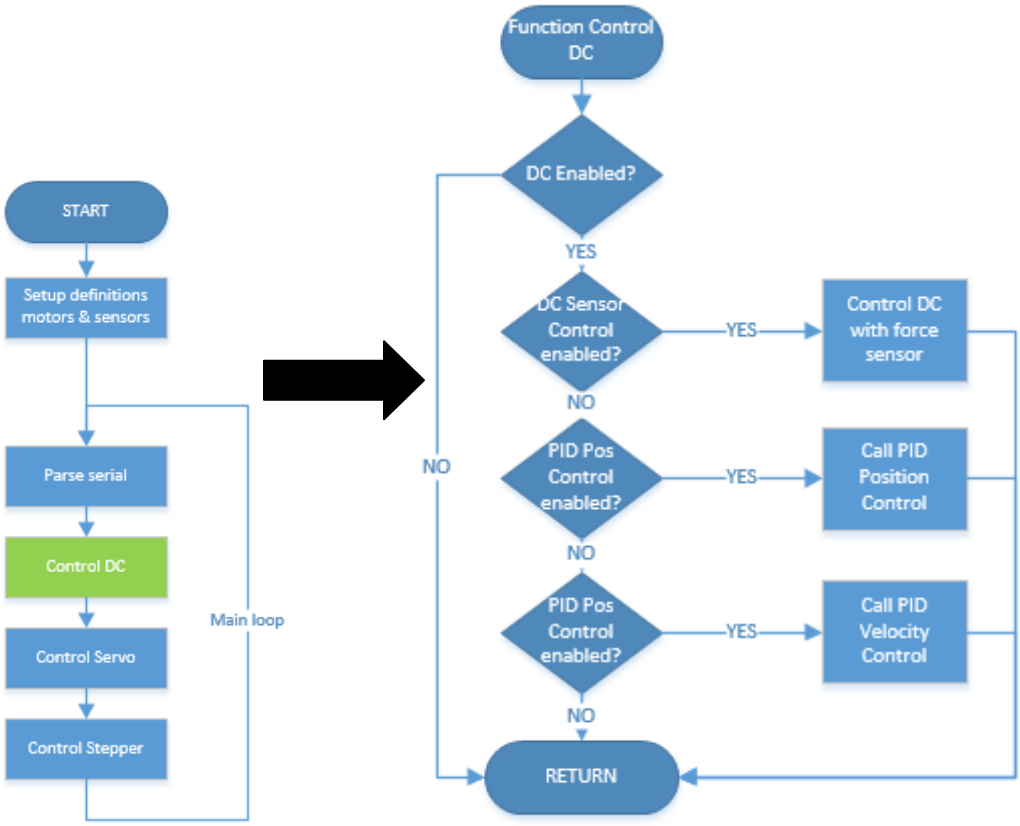

Figure 1.1: Flow Chart of the Project

1.1.1 Force Sensor

The force sensor was interfaced using Table 1.1 with the RAMBo board[1].

Table 1.1: Sensor Interface Pins with RAMBo

Off Board Circuit	Arduino Pin/ Connector Pin
10K ohm resistor end1	Gnd/ Ext2.1
10K ohm resistor end2	Pin59/ Analog Extension4
Force Sensor End1	5v/ Ext2.3
Force Sensor End2	Pin59/ Analog Extension4

The force sensor varies its resistance with as the force applied to it increases. This variation was mapped from resistance to a voltage variation by using the analog pin and the function analogread() on the Arduino and the code respectively. This was then constrained from 0 to 255 and applied as a PWM to the DC motor.

1.1.2 DC Geared Motor

The DC geared motor was controlled using a PWM for force sensor control and controlled using PID for position control and velocity control. Table 1.2 shows the interface pins for the DC motor with the RAMBo board. The first task that I completed was reading from encoders. The first implementation was to enable the pull up resistor present on the Arduino. Then the encoder channels A and B are read in the interrupt. Once, I implemented this I realized that the implementation was very slow which caused samples to be misread. Then as suggested by my teammate Ihsane, I used the encoder library[2] provided by Arduino to read these encoder values. After this I implemented PID algorithm for controlling the DC motor based on the target value set by the user. For this I used the PID library provided by Arduino which computes the PWM value for the given input. However, I do understand how these libraries are implemented, but they save time for implementation. TO power the motor we needed a 12V supply from the board which is supplied by the Heat-Bed connector of the board by enabling it. The tuning of the PID constants was tough. Finally my teammates Dan and Ihsane gave me the intuition on how to compute these constant which helped me calibrate the constants more easily. The learning was that we need to tune the PID constant first till the motor oscillates, then tune the PID differential constant such that the motor stops oscillating and then finally tune the integral constant to reduce the overshoot error.

1. Link to RAMBo 1.1v Manual: http://reprapelectro.com/wpcontent/uploads/2014/09/RAMBo-1.1B-User-Manual.pdf
2. Link to encoder library: http://www.pjrc.com/teensy/td_libs_Encoder.html

Table 1.2: DC Motor Interface Pins with RAMBo

Off Board Circuit	Arduino Pin/ Connector Pin
DC Motor M+	12v/Heat-Bed
DC Motor M-	Gnd/Heat-Bed
DC Motor V+	5v/ Ext2.3
DC Motor V-	Gnd/ Ext2.1
DC Motor ChannelA	79/Ext2.18
DC Motor ChannelB	80/Ext2.20

1.2 MRSD project

For the project ADD_IN, I was responsible to design and make a 3D model of the coupler that can help convert the nozzle extrusion plane to a plane that is at an angle of 45 degrees to it. To do this we first intended to make a new nozzle design. However, we realized it is easier to do test using a coupler and then I designed the coupler shown in Figure 1.2. The plane is changed by designing a curvature to rotate the plane of extrusion by 45 degrees. 45 degrees has been chosen because the nozzle should not hit the added COTS item or the plane of printing.

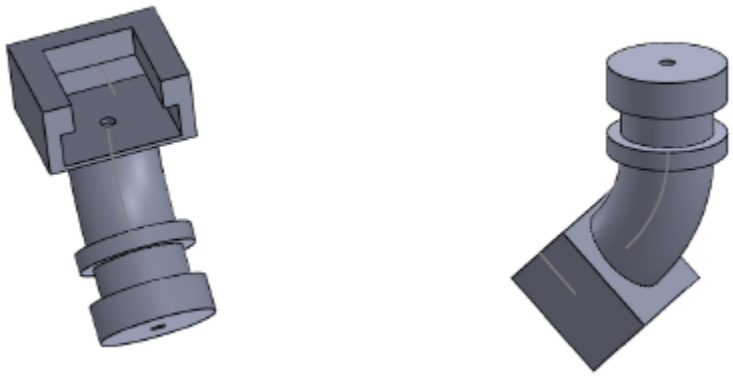

Figure 1.2: Nozzle CAD design 3D views

2 Challenges

The major challenge while implementing the sensors and motors project was in integration. Everyone had different coding techniques which led to a time costly integration. However, the pin assignment in the board was done well to ensure the team does not face problems while integration with respect to pins. To avoid this we have discussed coding standards which the team will adhere to in future.

---

1. Link to RAMBo 1.1v Manual: http://reprapelectro.com/wpcontent/uploads/2014/09/RAMBo-1.1B-User-Manual.pdf
2. Link to encoder library: http://www.pjrc.com/teensy/td_libs_Encoder.html

The second challenge that I faced was tuning with PID which my teammates helped me out with.

The third challenge I faced in the designing the part in solidworks, making it modular to make changes. To solve this I approached the TA Stephen and asked him general pointer which has really helped me.

3 Teamwork

- Astha: She worked on the control of the servo motor and interfaced it with a button switch and a sharp IR sensor. She played a major role in building the mechanical base, wiring and integrating.
- Dan: He worked on the GUI. He made an amazing code that was readable. Helped me with the PID tuning. He also played a major role in building the mechanical base, wiring and integrating. In the project he helped me think about the experiment that we might do to test kinking of filament.
- Ihsane: He played an important role in stepper motor design and interfacing it with the potentiometer. He also played a major role in building the mechanical base, wiring and integrating. His suggestions to use the library saved a lot of implementation time and improved the performance of the system.

4 Plan

Till the next demo I plan to work on printing the designed coupler and testing it. After this I intended to design the CAD file for a coupler which can incorporate various tubes at the centre of the coupler. This is to test the material properties of the filament with the nozzle which may be aluminum or different for different materials.

5 Code

```
//DC_Motor.h
#ifndef DCmotor
#define DCmotor

#define DEFAULT_DCMOTOR_GAIN 10
#define DEFAULT_TARGET_POSITION 10000
#define DEFAULT_TARGET_VELOCITY 10000
#define DEFAULT_SENSOR_CONTROL_MODE LOW
```

```
#define DEFAULT_PID_CONTROL_MODE LOW
#define Kc_p 1.3
#define Ki_p .18
#define Kd_p .2
#define Kc_v .005
#define Ki_v .0004
#define Kd_v .0008
#define MAX_PWM 255

void setupDCmotor();
void controlDCmotor();
void controlMotorPosition();
void controlMotorVelocity();

void setupForce();
void readForce();

#endif
```

```
//DC_Motor.ino

#include "pindefinitions.h"
#include "protocoldefinitions.h"
#include "DCmotor.h"
#include "main.h"
#include <Encoder.h>
#include <PID_v1.h>


extern stateVariables sv;

int previousForce = 0;
double Output_p, Output_v, targetDistance = 0;
unsigned long lastTime;
signed long lastBase = 0;
signed int v_pwm;

Encoder base(CHANNELA, CHANNELB);
PID position_PID(&sv.dcCurrentPos, &Output_p, &sv.dcTargetPos, Kc_p, Ki_p, Kd_p,
DIRECT);
PID velocity_PID(&sv.dcCurrentVelocity, &Output_v, &sv.dcTargetVelocity, Kc_v,
Ki_v, Kd_v, DIRECT);
int stepperSpeed = 10;

void setupDCmotor() {
  Set up pints
```

```
  pinMode(DC_ENABLE_PIN, OUTPUT);
  pinMode(MOTOR_SUPPLY, OUTPUT);
  pinMode(DIR1_PIN, OUTPUT);
  pinMode(DIR2_PIN, OUTPUT);
  pinMode(CHANNELA, INPUT);
  pinMode(CHANNELB, INPUT);
  digitalWrite(DC_ENABLE_PIN, LOW);
  digitalWrite(MOTOR_SUPPLY, HIGH);
  digitalWrite(DIR1_PIN, LOW);
  digitalWrite(DIR2_PIN, LOW);
  set up pull up resistors for encoder
  digitalWrite(CHANNELA, HIGH);
  digitalWrite(CHANNELB, HIGH);

  initialize variables
  sv.dcGain = DEFAULT_DCMOTOR_GAIN;
  sv.dcCurrentPos = 0;
  sv.dcCurrentVelocity = 0;
  sv.dcTargetPos = DEFAULT_TARGET_POSITION;
  sv.dcTargetVelocity = DEFAULT_TARGET_VELOCITY;
  sv.dcSensorControlMode = DEFAULT_SENSOR_CONTROL_MODE; true = sensor
control mode, false = GUI control mode
  sv.dcPositionPIDMode = DEFAULT_PID_CONTROL_MODE; true = position PID
control, false = velocity PID control
  sv.dcEnabled = LOW; true = motor enabled, false = motor disabled

  Position PID Settings
  position_PID.SetOutputLimits(-255,255);
  Velocity PID Settings
  velocity_PID.SetOutputLimits(-255,255);

  sv.buttonStatus = LOW;
  position_PID.SetMode(AUTOMATIC);
  velocity_PID.SetMode(AUTOMATIC);
  velocity_PID.SetSampleTime(20);
}


void enableDCmotor(bool enable) {
  sv.dcEnabled = enable;
}

255 = full forward
 0 = stop
-255 = full reverse
void runDCmotor(int pwm) {
```

```
  sv.dcPWM = pwm;
  if(pwm > 0)
  {
    analogWrite(DC_ENABLE_PIN, pwm);
    digitalWrite(DIR1_PIN, HIGH);
    digitalWrite(DIR2_PIN, LOW);
  }
  else{
    analogWrite(DC_ENABLE_PIN, pwm*-1);
    digitalWrite(DIR1_PIN, LOW);
    digitalWrite(DIR2_PIN, HIGH);
  }
}


void controlDCmotor() {
  int force = sv.dcSensorVal;
  if (sv.dcEnabled)
  {
    unsigned long now = millis(); Get current time
    signed long newbase = base.read(); get current encoder pos
    unsigned long timeChange = (unsigned long)(now - lastTime);
    sv.dcCurrentPos = (double)base.read() + 10000.0; update current position state
variable
    sv.dcCurrentVelocity = (double)(lastBase - newbase)/(double)timeChange; update
current velocity state variable
    sv.dcCurrentVelocity *= 10000.0;
    sv.dcCurrentVelocity += 10000.0;
    lastTime = now;
    lastBase = newbase;

    if (sv.dcSensorControlMode)
    {
      float gain = (float)sv.dcGain / 100.0;
      int16_t pwm = ((int16_t)sv.dcSensorVal) * gain;
      pwm = constrain(pwm, 0, 255);
      runDCmotor(pwm);
    }
    else {
      if (sv.dcPositionPIDMode) {
          controlMotorPosition();
      }
      else{
        controlMotorVelocity();
      }
    }
```

```
  }
  else{
    analogWrite(DC_ENABLE_PIN, 0); disable motor
  }

}

void controlMotorPosition() {
  position pid control code
  /*
  Serial.print(RESP_DEBUG);
  Serial.print("Output_p = ");
  Serial.println(Output_p);
  */
  position_PID.Compute();
  runDCmotor(Output_p);
}

void controlMotorVelocity() {
  //velocity pid control code
  /*Serial.print(RESP_DEBUG);
  Serial.print("error = ");
  Serial.println(sv.dcCurrentVelocity - sv.dcTargetVelocity);*/

  velocity_PID.Compute();

  v_pwm -= Output_v;
  v_pwm = constrain(v_pwm,-255,255);
  runDCmotor(v_pwm);
}

//////////**********FORCE**********/////////////////////////

void readForce() {
  sv.dcSensorVal = analogRead(FORCE_PIN);
}

void setupForce() {
  pinMode(FORCE_PIN, INPUT);
  sv.dcSensorVal = analogRead(FORCE_PIN);
}

//////////**********ENCODER**********/////////////////////////
```