Ihsane Debbache

Team F: ADD_IN

Teammates: Nikhil Baheti, Dan Berman and Astha Prasad

ILR01

October 16th, 2015

I - Individual progress

My part in this sensors lab was to interface a stepper motor with a potentiometer using a variable gain that can be set from the graphical user interface.

I - 1 - Hardware: For this implementation, I used the Rambo Board (RepRap Arduino-compatible Mother Board), which is a popular 3D printer controller board that is based on the Arduino Mega 2560 and offers more capabilities. Specifically for this lab, it has onboard A4988 stepper drivers as shown in figure 1. This is the board we are using in our MRSD project so it made sense to use it for this lab. The potentiometer I interfaced with the stepper is also shown in the figure.

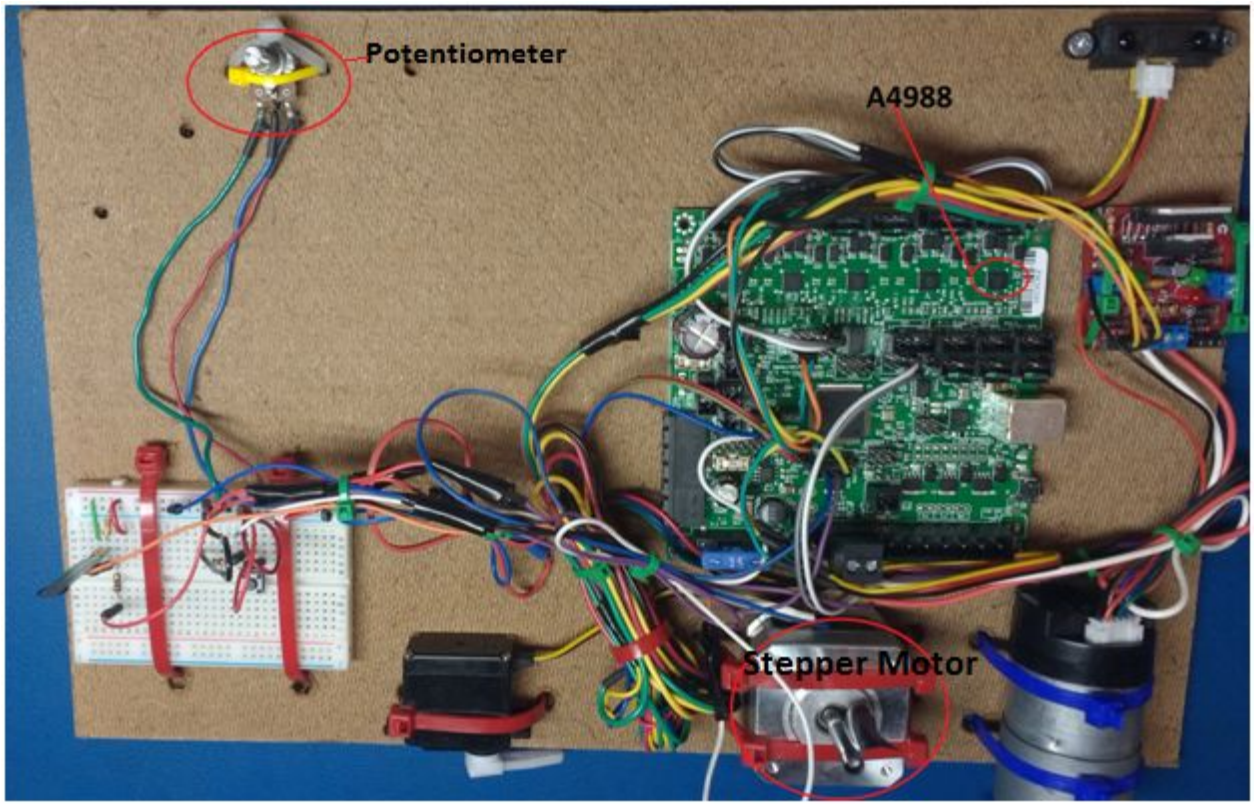
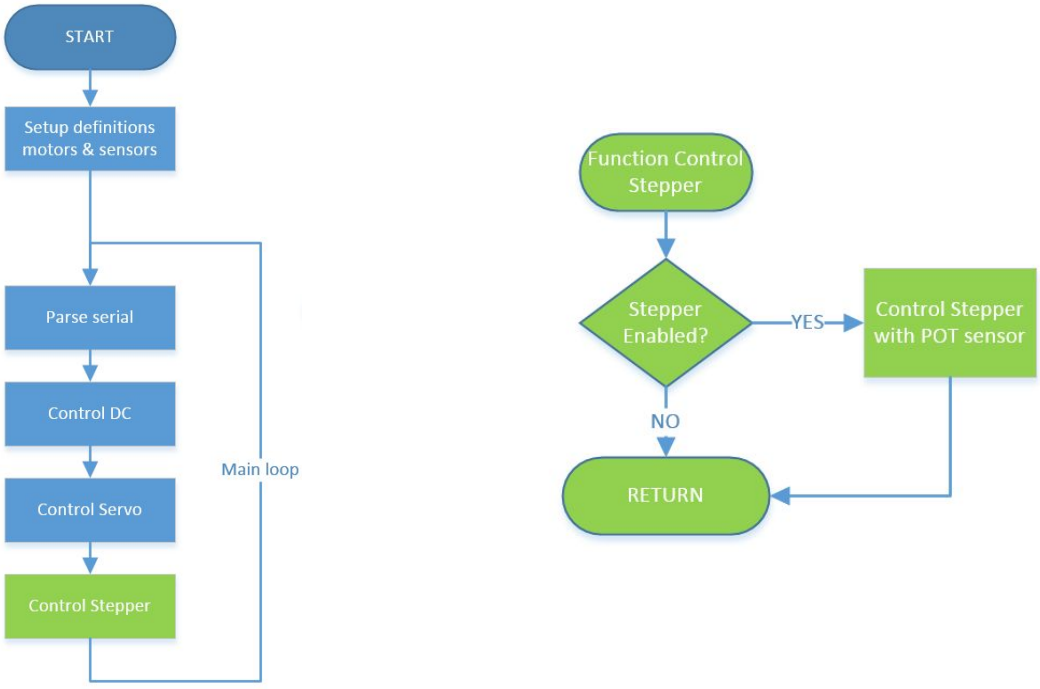


Figure 1: Circuit implementation highlighting the Stepper, potentiometer and A4988 driver

The hardware implementation of this part was fairly straight forward, the stepper was connected to one of the Rambo board's motor ports, and the potentiometer output pin to an extension digital pin on the board. The rest of the connections, like from the A4988 to the motor power source and to the microcontroller are on the board, and can be set in the software using the correct pins. The board's manual was greatly helpful with that

I - 2 - Software: My part in the software is highlighted in green on the simplified flowchart in Figure 2 below:



Before interfacing the stepper to the potentiometer, I tested each separately. First using a simple code to read and print the sensor value of the potentiometer. The readings were effectively reading 0 at one limit and 1024 at the other, which maps to potentiometer voltages of 0V and 5V. From that it was fair to assume a linear relationship between potentiometer knob angle and output voltage. Second, I wrote a function to move the stepper by a set number of steps in a given direction at a given speed, leaving the micro stepping settings fixed. The whole code for this part is in Appendix A.

II - Challenges

This part was fairly simple compared to the others, but the biggest challenge was integrating the code with the rest. Since I had limited experience in programming, my first code was at written in the setup and loop, without following the coding standards, and we also did not have a clear flowchart highlighting each function. So after I tested the code separately, when time came to intergrate it I had to rewrite it as functions and definitions, which took more time than doing the first code since it was new to me and Dan greatly helped with that.

III - Teamwork

We divided the tasks as instructed, with one member doing the GUI and each of the other three members interfacing a motor with a sensor: So Astha Prasad interfaced the proximity sensor with a servo motor. Nikhil Baheti implemented PID position and velocity control on the DC motor and also interfaced it with a force sensor, and Dan Berman wrote the GUI and greatly helped in integrating the parts together. We tried to divide the work as evenly as possible and in a way that made sense in relation to our roles in the project. Although it is fair to mention that the GUI represented a disproportionally large workload compared to the rest. Dan, who has the best programming experience among us, had to wrote over 600 lines of code for it, and then helped us integrate the parts together and with the GUI.

IV - Future Plans

My focus in the project, along with Nikhil's, is designing the 45° bent Nozzle that is capable of rotating about its Z axis for our 3D printer. We identified some risks in this part, mainly with respect to the kinking of the filament due to the bent and rotation. So my current task this week are 3D printing and testing the extruder to nozzle connectors shown in figure 3, that we will use to experiment and validate the feasibility of our current method.

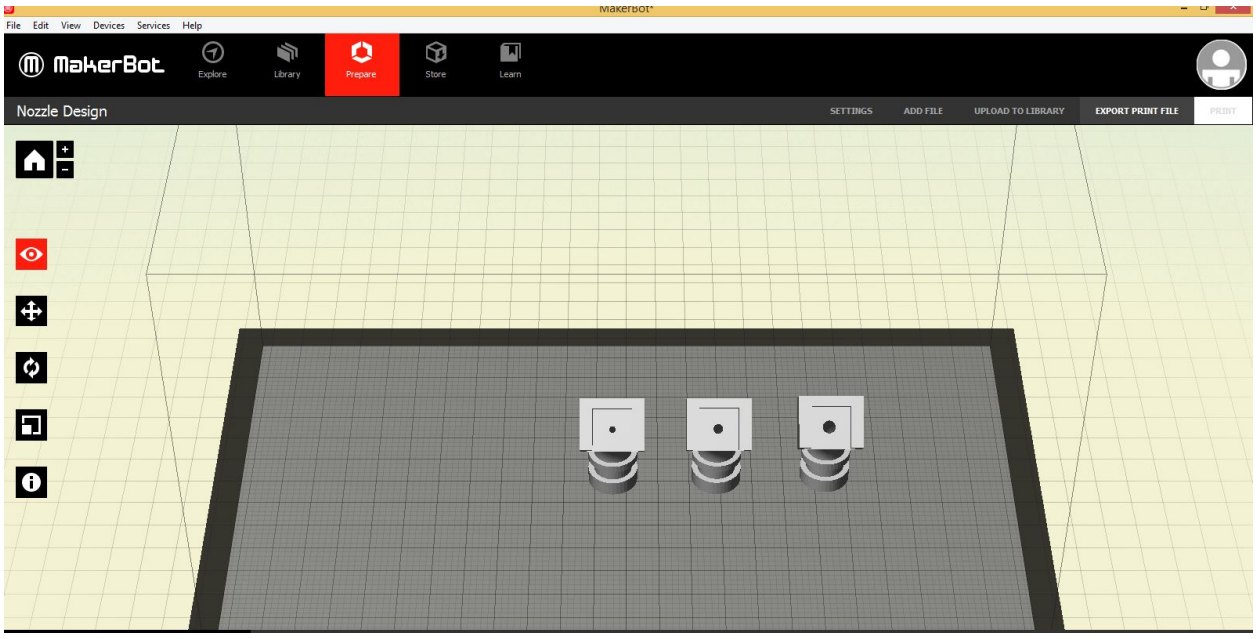


Figure 3: Bent extruder to nozzle connectors with various tube diameters.

I will also be studying nozzle designs and working on identifying the most appropriate of the shelf rotary stages for our project, using parameters such as precision, speed, weight. repeatability and cost. I will then conduct a trade study to identify the best one.

Appendix A:

Below is the code used to control the stepper with the potentiometer:

```
#include "pindefinitions.h"
#include "protocoldefinitions.h"
#include "stepper.h"
#include "main.h"



extern stateVariables sv;

int stepperSpeed = 10;

void setupStepper(){
  //Set up pints
  pinMode(STEP_PIN, OUTPUT);
  pinMode(DIR_PIN, OUTPUT);
  digitalWrite(STEP_PIN, LOW);
  digitalWrite(DIR_PIN, LOW);
  pinMode(STEPPER_ENABLE_PIN, OUTPUT);
  pinMode(XMS1, OUTPUT);
  pinMode(XMS2, OUTPUT);
  digitalWrite(XMS1, LOW);
  digitalWrite(XMS2, HIGH);

  sv.stepperCurrentPos = 32767;
  sv.stepperGain = DEFAULT_STEPPER_GAIN;
  stepperEnable(false);
}

void stepperhome(){
  sv.stepperCurrentPos = 0;
}

// A custom delay function used in the run()-method
void holdHalfCycle(double speedRPS) {
  long holdTime_us = (long)(1.0 / (double) stepsInFullRound / speedRPS /
2.0 * 1E6);
  int overflowCount = holdTime_us / 65535;
  for (int i = 0; i < overflowCount; i++) {
    delayMicroseconds(65535);
  }
  delayMicroseconds((unsigned int) holdTime_us);
}

// Runs the motor according to a chosen direction, speed (rounds per
seconds) and the number of steps
void runStepper(boolean runForward, double speedRPS, int stepCount) {
  digitalWrite(DIR_PIN, runForward);
  for (int i = 0; i < stepCount; i++) {
    digitalWrite(STEP_PIN, HIGH);
```

```
    holdHalfCycle(speedRPS);
    digitalWrite(STEP_PIN, LOW);
    holdHalfCycle(speedRPS);
  }
  if(runForward)
    sv.stepperCurrentPos -= stepCount;
  else
    sv.stepperCurrentPos += stepCount;
  }

void controlStepper(){
  int pot = sv.stepperSensorVal;
  if(sv.stepperEnabled)
  {
      float gain = (float)sv.stepperGain/100.0;
      int16_t targetPos = ((int16_t)pot-512)*gain + 32767;
      int16_t error = sv.stepperCurrentPos - targetPos;
    if(error > 0)
       runStepper(true, stepperSpeed, error);
    else
       runStepper(false, stepperSpeed, error*-1);
  }
}

void stepperEnable(bool enable){
    digitalWrite(STEPPER_ENABLE_PIN, !enable);
    sv.stepperEnabled = enable;
}

//////////***********POT**********////////////////////////////

void readPot(){
    sv.stepperSensorVal = analogRead(POT_PIN);
}

void setupPot(){
  pinMode(POT_PIN, INPUT);
   sv.stepperSensorVal =0;
}
```