**PS: Electronics design for the actuation of a 3D-printed robot arm with 500 mm reach and maximum 1 kg payload.**

**Week 5**

Previous week’s suggestions:

* To start doing work on the electronics design of the robot.

Initial design:

MATLAB has been used by the software team for the implementation of the robot control logic. Since the Arduino environment can be integrated with MATLAB and has libraries for controlling servos, as well as for its ease of use, it was chosen to be used as the robot controller. MATLAB can be used to code the Arduino directly using the “MATLAB Support Package for Arduino Hardware” add-on. The strategy was to do all the processing for the robot logic in MATLAB in the connected PC itself, calculate the required servo angles and then direct the servos using the Arduino.

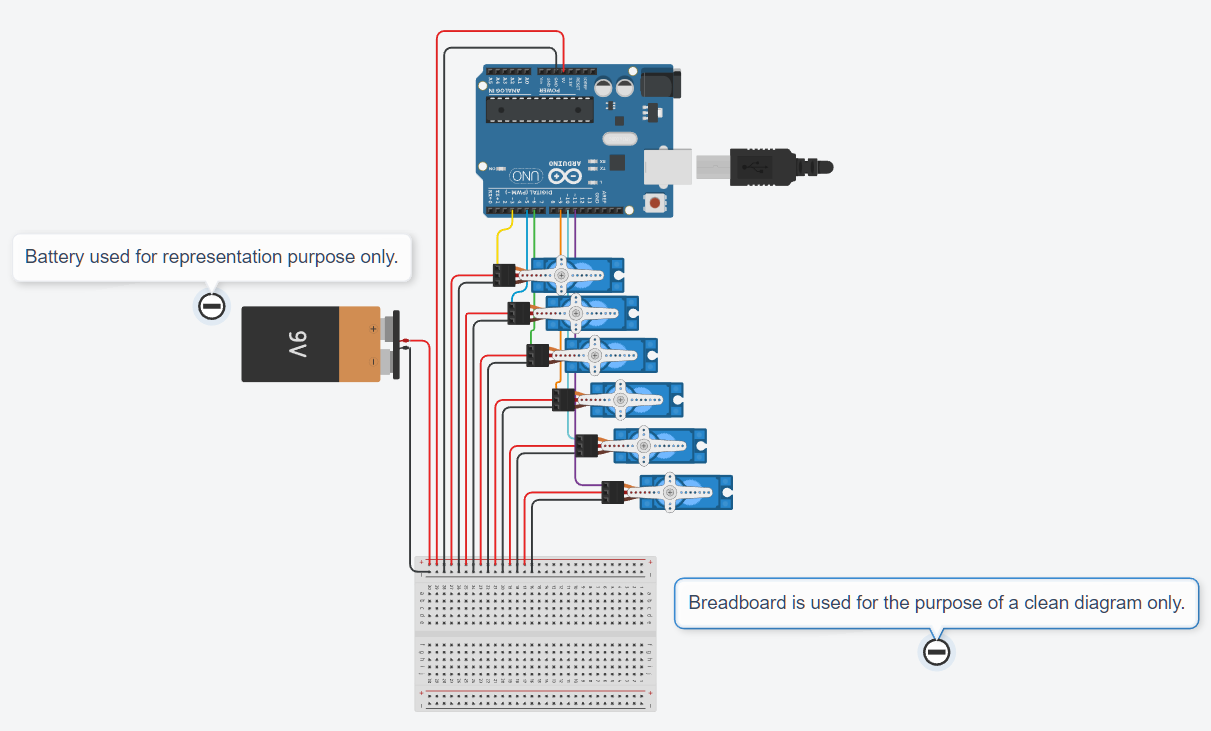


Fig 1: Initial circuit diagram containing the Arduino connected to the 6 servos through the PWM enabled pins.

The power requirements of the servos have been extracted from their datasheets below:

|  |  |  |  |
| --- | --- | --- | --- |
| Servo | Operating Voltage | Stall Current (each) | Peak Stall Current (total) |
| 4 x 35 kg cm (Model no. unknown) | 7.2 - 8.4V | 3.2 -3.5A | 14A |
| 2 x 80 kg cm (DS5180) | 6 - 7.4 - 8.4V | 3.5 - 5.6 - 6.2 A | 12.4A |
|  |  |  | 26.4A |

Hence, an 8V-30A-240W power supply would be suitable.

Servo Power Supply Circuit:

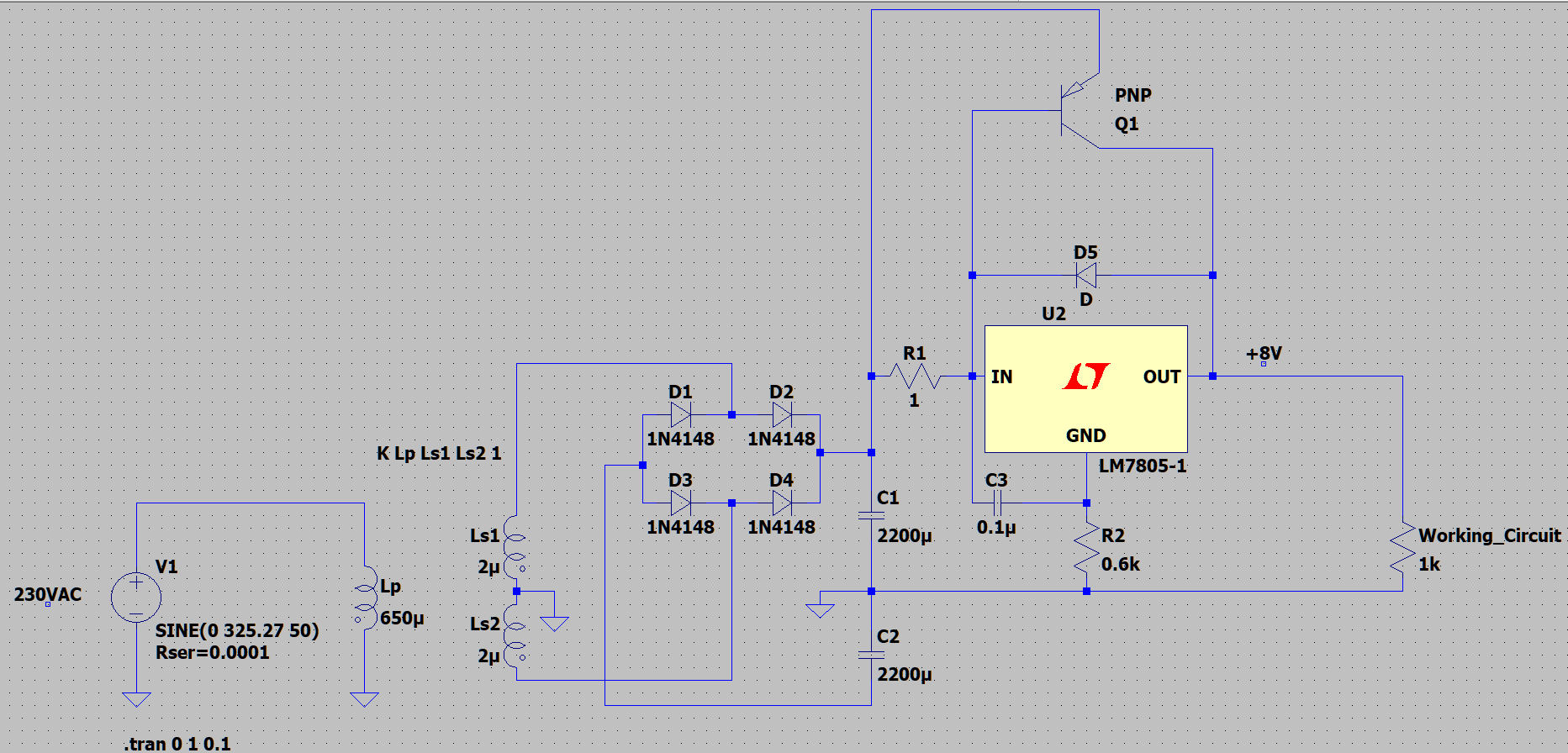


Fig 2: LtSpice circuit diagram of the servo power supply.

**Week 6**

Previous week’s suggestions:

* To develop the required servo control code (for translating angles to servo input signals, among other things) for the newly designed controller.
* The power supply will be procured from external source and hence need not be designed.

Progress in this week:

* The servo control circuit has been designed.
* A simulation of the circuit using Arduino has been done in Proteus.

Servo control:

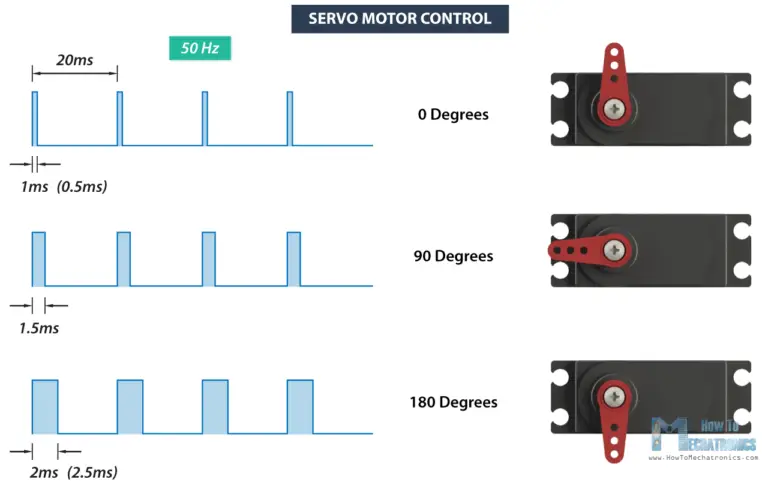
****

Fig 3: Servo motor control

Hence,

x deg -> {(100/9) + 500} us

x us -> {(x - 500) \* 0.09} deg

Control Circuit:

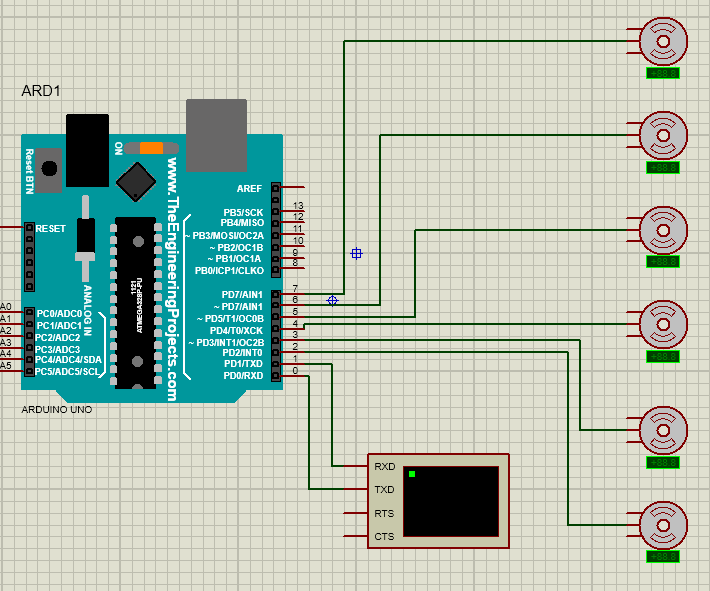


Fig 4: Servo controller circuit diagram in Proteus.

Controller code:

#include <Servo.h>

Servo servo1, servo2, servo3, servo4, servo5, servo6; // create servo object to control a servo

unsigned int servopos[6] = {500, 500, 500, 500, 500, 500} ; // uint to hold servo position in microseconds (0-180 deg is mapped to to 500-2500 us), initialised to 0 deg

String inString = ""; // string to hold input

int counter = 0; //Counter for the servopos array

void setup() {

servo1.attach(2,500,2500); // (pin, min angle us, max angle us)

servo2.attach(3,500,2500);

servo3.attach(4,500,2500);

servo4.attach(5,500,2500);

servo5.attach(6,500,2500);

servo6.attach(7,500,2500);

// Initialise all servos to 0 deg

servo1.writeMicroseconds(servopos[0]);

servo2.writeMicroseconds(servopos[1]);

servo3.writeMicroseconds(servopos[2]);

servo4.writeMicroseconds(servopos[3]);

servo5.writeMicroseconds(servopos[4]);

servo6.writeMicroseconds(servopos[5]);

Serial.begin(9600); // Start the serial communication

}

void loop() {

// Read serial input:

while (Serial.available() > 0) {

int inChar = Serial.read();

if (isDigit(inChar)) {

// convert the incoming byte to a char and add it to the string:

inString += (char)inChar;

}

if (inChar == 'p') {

counter++;

}

// if we get an underscore, store the int value in the respective servopos element:

if (inChar == '\_') {

servopos[counter] = inString.toInt();

counter++;

// clear the string for new input:

inString = "";

}

if (inChar == 'e') {

// Write the new servo positions

servo1.writeMicroseconds(servopos[0]);

servo2.writeMicroseconds(servopos[1]);

servo3.writeMicroseconds(servopos[2]);

servo4.writeMicroseconds(servopos[3]);

servo5.writeMicroseconds(servopos[4]);

servo6.writeMicroseconds(servopos[5]);

// Reset the counter

counter = 0;

}

}

}

The controller first initializes all the servos to 0 degrees. Then it awaits commands via the serial receiver. A custom command sequence has been developed in which the pulse durations are input in microseconds in order, separated by underscores (‘\_’) and the end of the command is indicated with the letter ‘e’. If no change from the previous input is desired, the pulse duration and underscore for the corresponding servo can be replaced by inputting the letter ‘p’ instead.

Example commands:

* 1500\_500\_500\_500\_2500\_700\_e
  + Servo 1 - 90 deg
  + Servo 2 - 0 deg
  + Servo 3 - 0 deg
  + Servo 4 - 0 deg
  + Servo 5 - 180 deg
  + Servo 6 – 18 deg

**Week 7**

Previous week’s suggestions:

* To add the required components in the electronics design to the BOM.

Progress in this week:

* The BOM has been updated.
* The code to convert the calculated angles to microseconds, format the pulse duration data into the custom form and then transmit it via the serial port has been completed.