Project: Baby Segway

Name: Zhao Yichun School: Victoria Junior College

Isha Singh Le Xue Anglo-Chinese Junior College

**I. Basics of Segway**

Segway is a two-wheeled vehicle designed to be self-balancing. When it is pushed, it moves in the direction of the force applied. Using this mechanism, the user can control the segway by leaning forward or backwards to make it move in the same directions. A handlebar is also installed for the user to turn by changing the speed of one wheel.

In order to detect whether the Segway should move forward or backwards to balance itself, the gyroscope and accelerometer are used. The gyroscope measures the angular speed of the movement while the accelerometer measures the acceleration with respect to the Earth. It is also known as the gravity which provides information on the tilt of the vehicle. Based on these data collected, the Segway is programmed to respond in order to balance.

In this project, we made a “Baby Segway”, a mini self-balancing robot without any steering using the two sensors, motors and different softwares such as Arduino for coding the Baby Segway, and LibreCAD for designing to laser-cut the robot.

**II. Programming**

Our coding was done on the Arduino IDE software in order to upload it onto the ATMEGA328 microcontroller installed on the Arduino UNO board. This software allows us to input codes to control the motors based on the data received from both the gyroscope and the accelerometer.

|  |
| --- |
| int ax = A0;  int ay = A1;  int az = A2; |
| // naming accelerometer values collected for axis x, y and z and assigning pins    Connect analog pin 0, 1 and 2 to the x, y, and z pins of the accelerometer respectively.  GND is connected to the ground of the Arduino UNO board and VCC to 5V. |
| #include <L3G4200D.h>  #include <Wire.h> |
| // using libraries to control gyroscope    Based on the coding of the library is being used here, SDA is connected to analog pin 4 and SCL is connected to analog pin 5. GND is connected to the ground, and VCC is connected to 5V. |
| int L3G4200D\_Address = 105;  L3G4200D gyro; |
| // choosing gyroscope settings and selecting the enable gyroscope setting with the address |
| int gx;  int gy;  int gz; |
| // naming gyroscope values collected for axis x, y and z. |

|  |
| --- |
| H-Bridge.png |
| In order to get the H-bridge to work:  Vcc1 is connected to the power supply from the microcontroller.  Vcc2 is connected to an external power supply for the motors.  The negative end of the battery is connected to the ground.  At least one ground of the H-bridge is connected to the ground of the microcontroller.  EN - enabler. For 1A, 1Y, 2A and 2Y to work, the enabler must be connected to a HIGH pin.  A - control. For their respective Y’s to be HIGH, they must also be connected to a HIGH pin.  Y - output. The wire connected to what the H-bridge is to control, in this case, the motors.  Purpose of using H-bridge:  It allows us to control the motors to turn in both directions by the A and EN functions, and regulate the current to a value that suits the motors. |
| int enablePin1 = 9;  int foremotor1 = 3;  int backmotor1 = 4; |
| // assigning and naming digital pin for one side of H-bridge connected to first (left) motor  Connect digital pin 3 to 1A and digital pin 4 to 2A.  Then connect the front of the left motor to 1Y and the back of the left motor to 2Y. |
| int enablePin2 = 10;  int foremotor2 = 5;  int backmotor2 = 6; |
| // assigning and naming digital pin for other side of H-bridge connected to second (right) motor  Connect digital pin 5 to a 4A and digital pin 6 to 3A.  Then, connect the front of the right motor to 4Y and the back of the left motor to 3Y. |

|  |
| --- |
| void setup() { |
| Wire.begin();  gyro.initialize(2000); |
| // start up gyroscope  The gyroscope (L3G4200D) has three selectable scale - 250, 500 or 2000 dps (degrees per second). For this project, we have selected 2000 dps. |
| Serial.begin (9600); |
| // collect values from both gyroscope and accelerometer at the rate 9600 bps (bits per second) |
| pinMode(ax, INPUT);  pinMode(ay, INPUT);  pinMode(az, INPUT); |
| //setting accelerometer pins as input to collect data |
| pinMode (enablePin1, OUTPUT);  pinMode (foremotor1, OUTPUT);  pinMode (backmotor1, OUTPUT);  pinMode (enablePin2, OUTPUT);  pinMode (foremotor2, OUTPUT);  pinMode (backmotor2, OUTPUT);  } |
| // setting the H-bridge as output to power the motors |

|  |
| --- |
| void loop() { |
| Serial.print("AX = ");  Serial.print(analogRead(ax)-261);  Serial.print(" AY = ");  Serial.print(analogRead(ay)-331);  Serial.print(" AZ = ");  Serial.println(analogRead(az)-350);  delay(100); |
| // Print adjusted values of x, y and z of the accelerometer every 100ms on the same line  As the values of ax, ay and az are around 261, 331 and 350 when help upright respectively, we chose to minus those constant from the values so that we would usually have a value of 0 for all readings. This makes reading data collected easier.  “println” indicates to enter to the next line. This is for easier reading so that our gyroscope readings will not be on the same line as the accelerometer readings. |
| gx = gyro.getX();  Serial.print("GX:");  Serial.print(gx);  gy = gyro.getY();  Serial.print(" GY:");  Serial.print(gy);  gz = gyro.getZ();  Serial.print(" GZ:");  Serial.println(gz);  Serial.println(" ");  delay(100); |
| // Get x, y, z values from gyroscope and print values every 100ms on the same line.  Extra “Serial.println” leaves a line between values collected at different time, for simpler reading. |
| int v = analogRead(az)-350; |
| // let v be the adjusted reading for z-axis of accelerometer  This makes it easier for us to code our next functions without re-writing “analogRead(az)-350”. |
| if (v > 15 || gy < -100) |
| // when the accelerometer is tilted forward or the gyroscope is moved quickly forward |
| {  digitalWrite (enablePin1, HIGH);  digitalWrite (enablePin2, HIGH); |
| // enable both motors |
| digitalWrite (foremotor1, HIGH);  digitalWrite (backmotor1, LOW);  digitalWrite (foremotor2, HIGH);  digitalWrite (backmotor2, LOW);  } |
| // electricity runs through the pins in a way that allows the first motor to move forward  // electricity runs through the pins in a way that allows the second motor to move forward  With both enablers set to HIGH, electricity runs from 1Y or the front of the left motor (since 1A is HIGH) to 2Y or the back of the left motor (since 2A is LOW) and from 4Y or the front of the right motor (since 4A is HIGH) to 3Y or the back of the right motor (since 3A is LOW). This allows the motors to turn forward. |
| else if (v < -15 || gy > 100) |
| // when the accelerometer is tilted backwards or the gyroscope is moved quickly backwards |
| {  digitalWrite (enablePin1, HIGH);  digitalWrite (enablePin2, HIGH);  digitalWrite (backmotor1, HIGH);  digitalWrite (foremotor1, LOW);  digitalWrite (backmotor2, HIGH);  digitalWrite (foremotor2, LOW);  } |
| // enable both motors  // electricity runs through pins in a way that allows first (left) motor to move backwards  // electricity runs through pins in a way that allows second (right) motor to move backwards |
| else |
| // when the accelerometer and gyroscope are both stable |
| {  digitalWrite (enablePin1, LOW);  digitalWrite (enablePin2, LOW); |
| // disable both motors |
| digitalWrite (backmotor1, LOW);  digitalWrite (backmotor2, LOW);  digitalWrite (foremotor1, LOW);  digitalWrite (foremotor2, LOW);  } |
| // keep all pins disabled for the robot to balance and stay still |
| } |

**III. Building**

List of hardwares:

1 Arduino UNO

1 ATMEGA328 microcontroller

1 LM7805 voltage regulator

1 ADXL335 accelerometer

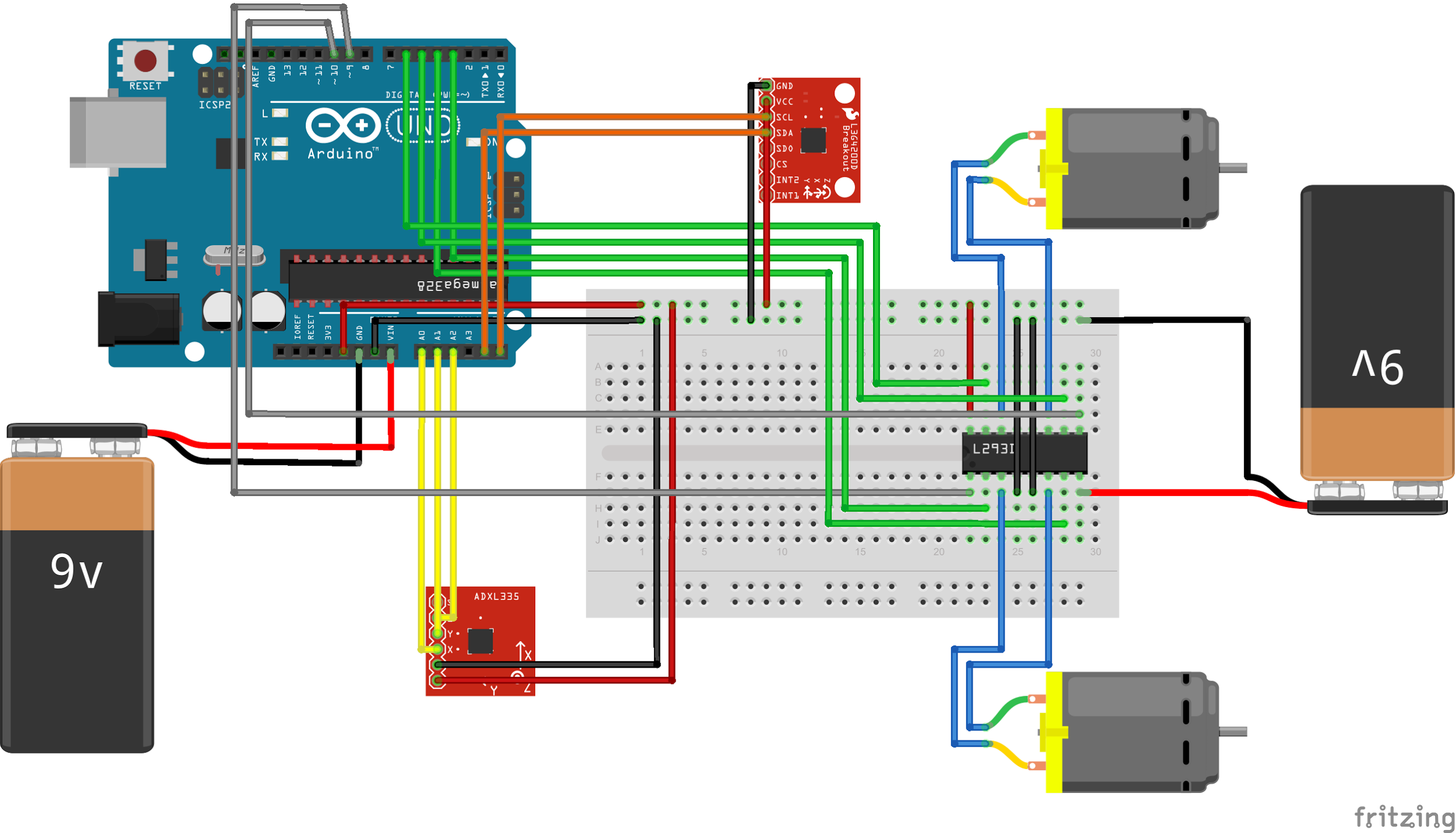
1 L3G4200D gyroscope

1 L293D H-bridge

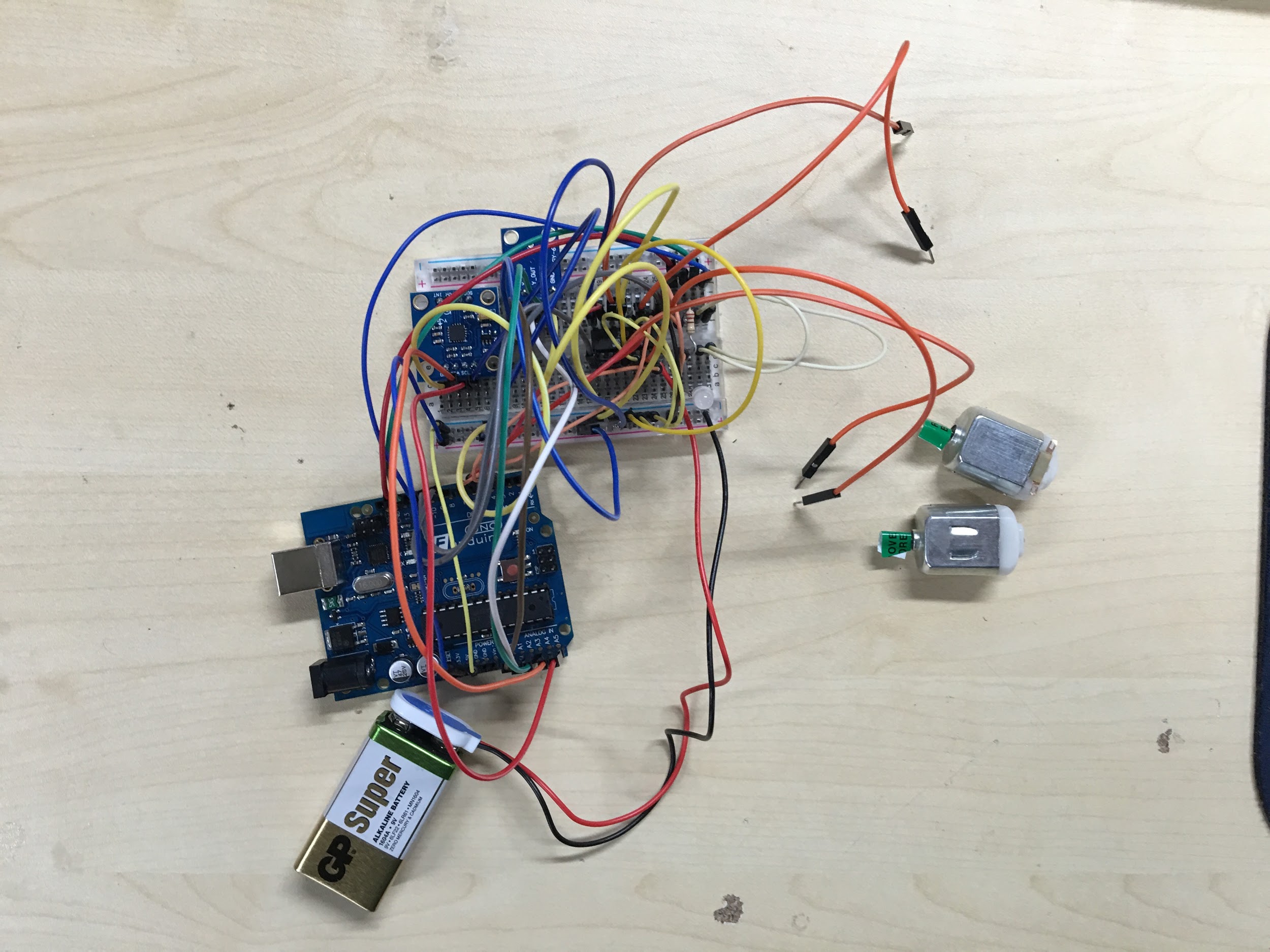
2 motors

2 wheels

Based on the above coding, the circuit would have to be connected as follows.



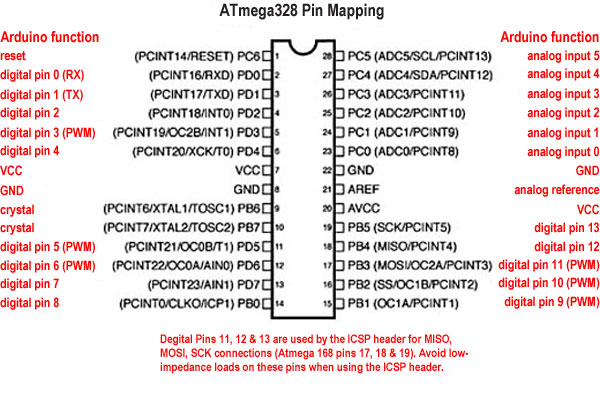
Below is a diagram of how it looks when connected using a breadboard:



**IV. Prototyping**

First, we had to ensure that our coding was working. This video in the [link (https://drive.google.com/file/d/0B5n-JJlw6XFKcU1TdHNxSkRaXzg/view?usp=sharing)](https://drive.google.com/file/d/0B5n-JJlw6XFKcU1TdHNxSkRaXzg/view?usp=sharing) is a prototype on our breadboard showing how the motors would move when the sensors on the breadboard were tilted.

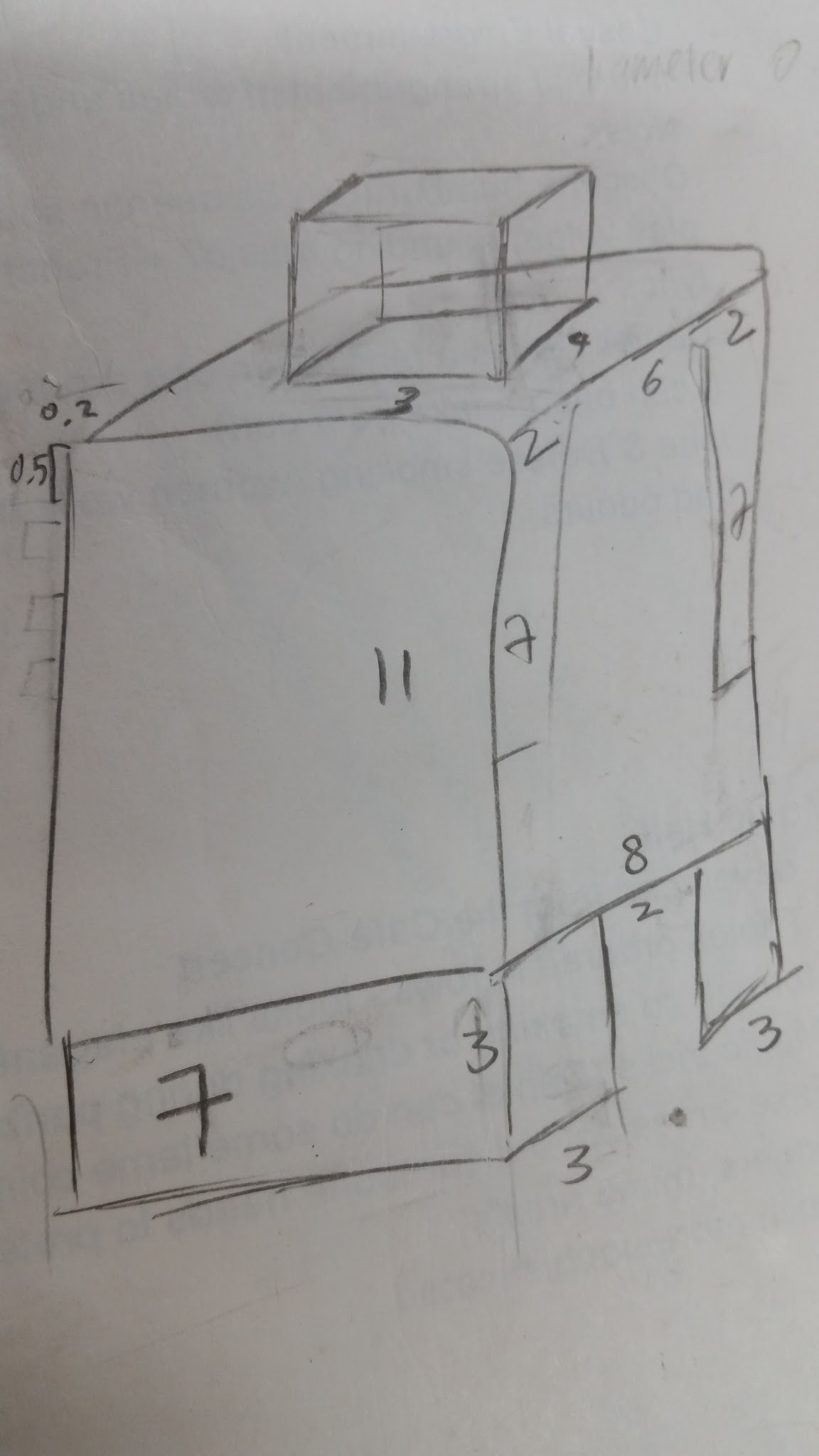
In order to test the values and ensure the circuit will work without the Arduino UNO board, we removed the microcontroller and placed it on the circuit board, and connected the circuit to the legs of the microcontroller as follows:



We then tested the values to decide what values should be used, to ensure the motors only move when the Segway is tilted to a certain extent. This was done by using a prototype that we had laser-cut using the LibreCAD software. It was designed to be slightly larger than the actual product as the breadboard and the Arduino UNO board needed to fit inside. For our final product, we designed a smaller and slimmer version.

Below is a draft of our prototype along with a picture of it.

Prototype draft:



Prototype:

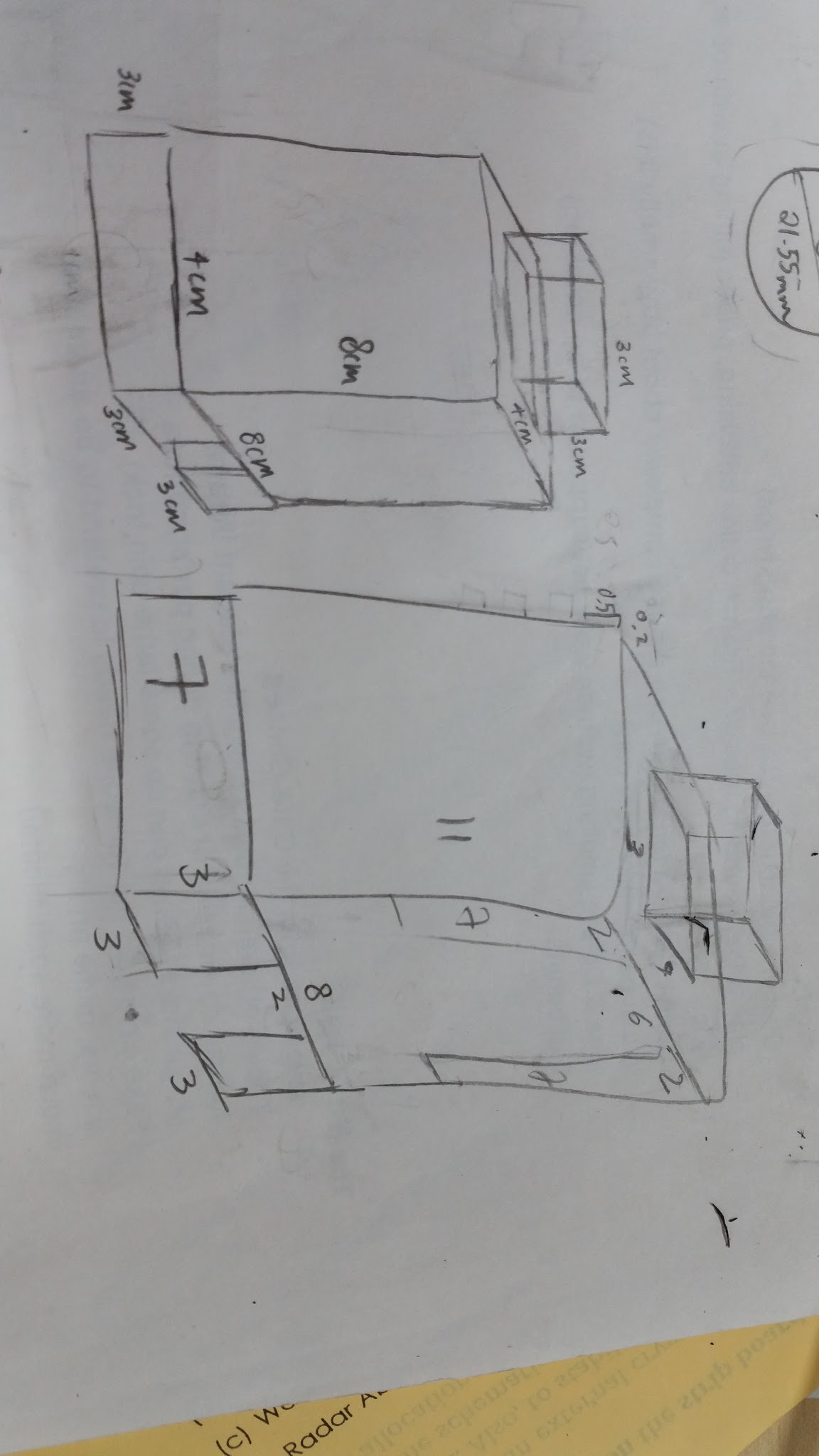


**V. Final Product**

After we had tested the value and ensured our circuits were working, we started to solder different components together.

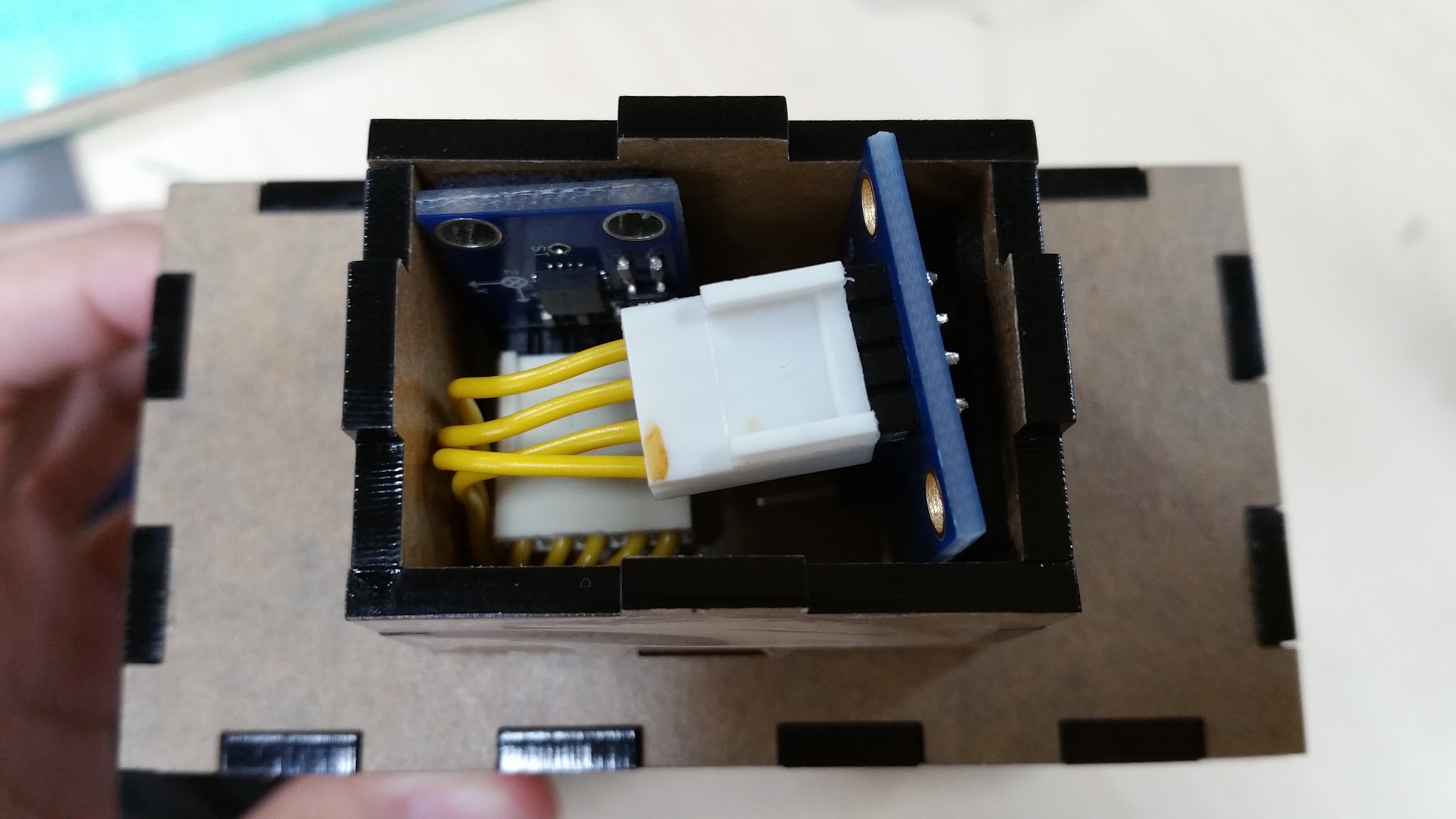
Our robot was designed into different compartments: head for the sensors, legs for the motors, and body for the circuit board and batteries.

Here is a draft of the final design:

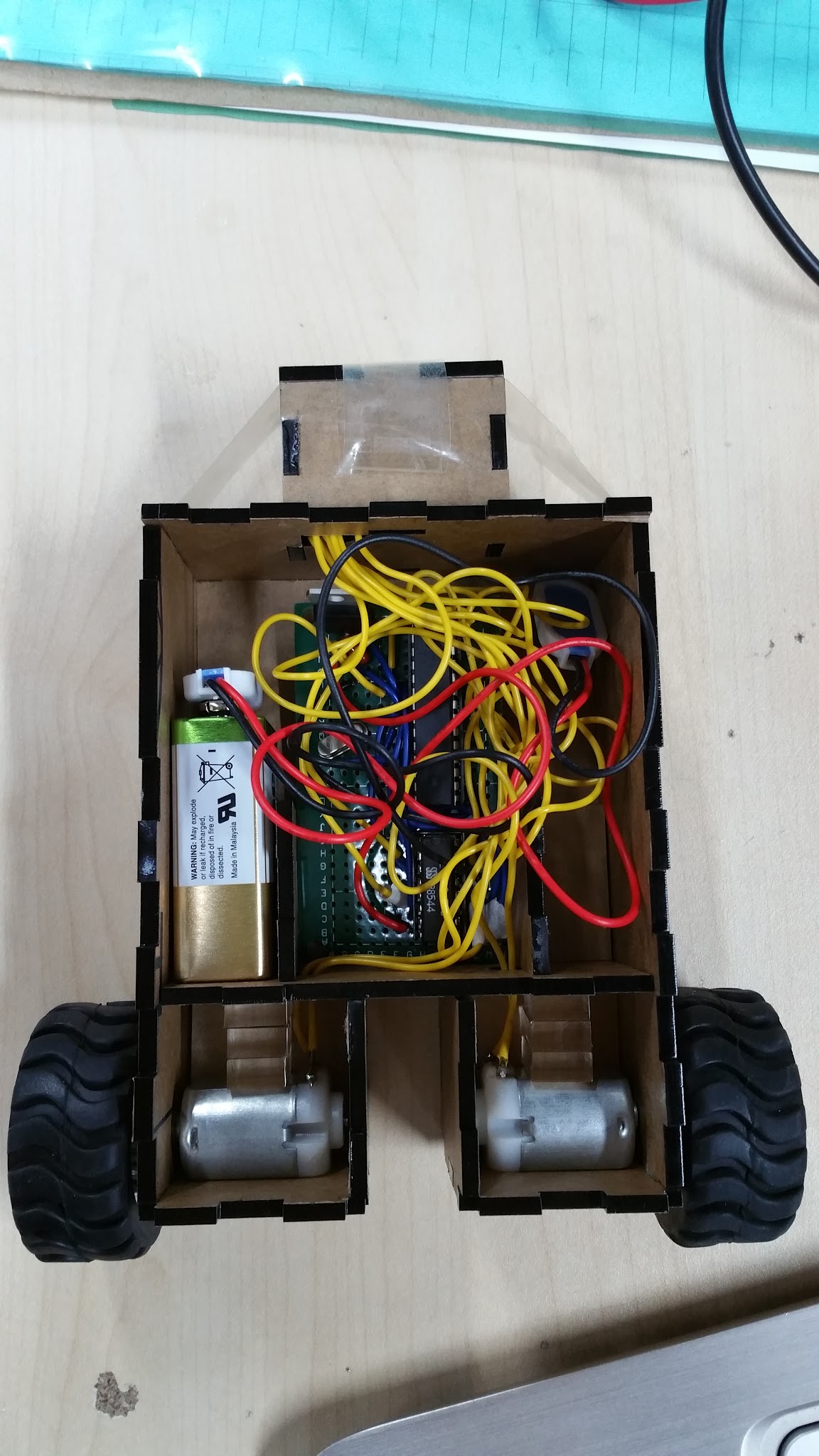


Below are how the compartments are connected.

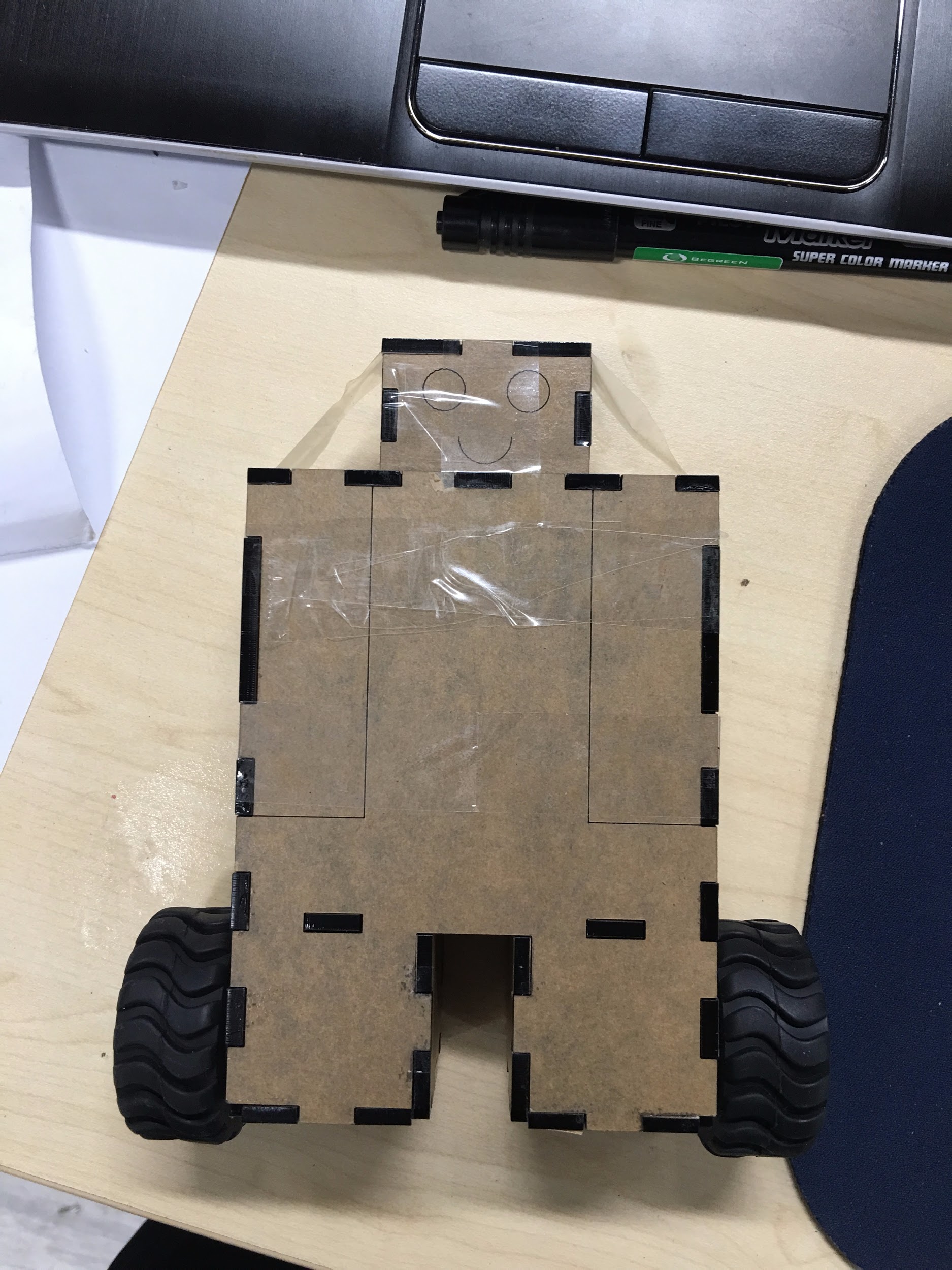
Head (gyroscope and accelerometer):



Body (circuit board and batteries) and Legs (motors):



Overall design:



Unfortunately, the motors often stopped running after a minute or so. Even when they were running at their strongest, they were still too weak to power the wheels to carry the robot, possibly because of the weight of the robot. However, we managed to make it work according to the codes. The [video here (https://drive.google.com/file/d/0B5n-JJlw6XFKMDZJSlBFSGM3Mm8/view?usp=docslist\_api)](https://drive.google.com/file/d/0B5n-JJlw6XFKMDZJSlBFSGM3Mm8/view?usp=docslist_api) displays how our Baby Segway worked.

Besides the problem with the weak motors, there are other areas of the project that we feel could have be improved on. Firstly, we can improve our Segway by controlling the speed of the motors when the tilt is smaller or larger. In this way, it will not move too much and have to continuously steady itself excessively. Secondly, it would be better to programme and make a mechanism for the Segway to steer, allowing it to turn left or right by having the left and right wheels have different speeds. This could have been done using the potentiometer.

Although our final product is not successful, we learnt a lot of things in the two weeks. The fact that we got our prototype to work is something that we are really proud of, because two weeks ago, we had no knowledge about Segway, microcontroller and programming. It has been fun working with our partners, trying to find out how things work and designing the prototype together. We would also like to thank our mentor, Mr Wong, for his helping hands through our learning journey.