ES2C6 Design Project

Self-Balancing Robot

Group 3

Rohan Dhir (U1839653)
Yasser Qureshi (U1839729)
Miles Mpofu (U1839740)
Ruoheng Wang (U1839858)
Pardeep Panesar (U1830518)
Michal Nahlik (U1830311)

School of Engineering,
University of Warwick,
Library Road
Coventry
CV4 7AL

December 2019

Table of Contents

1. Introduction

1.1 Mandate

Our task: design a low cost self-balancing robot which met the following criteria:

1. The ability to balance on both wheels for a minimum of 10 seconds
2. The ability to be fit in our storage box (dimensions 105 x 170 x 295mm) with the wheels removed
3. Can be safely deployed (i.e. not to cause harm by being unstable)
4. Not include any Arduino libraries

1.2 Group Aims

1. Innovative design: build on previous successes, adding new features, design and code improvements.

2. Use of Theory: learnt in lectures and additional theories such as the 'inverted pendulum'.

2. Theory

2.1 Inverted Pendulum

Our robot was designed with an 'Inverted Pendulum' in mind. This describes a system, where an inverted pendulum has to be maintained upright by moving the cart under its centre of mass.

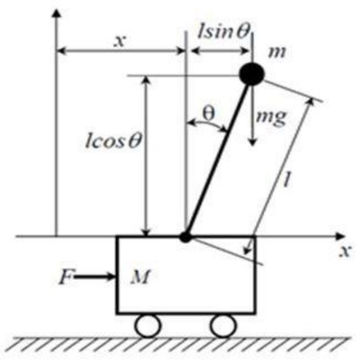


Figure 1 Theoretical diagram of Self-Balancing Robot

In this system, the pendulum has its centre of mass above the pivot point. The system is naturally unstable and without any help will fall over. In our case the system is a large mass at the end of a pole. The pole is free to rotate around its base, which itself moves in a horizontal plane, perpendicular to the vertical plane of the mass and pole. In the system above, when the pole is offset from its centre point, for example to the left of it, the cart must move to the left to re-centre the centre of mass.

To achieve a state of equilibrium we must control and stabilize the direction and acceleration of the robot. The motion of the robot will depend on the direction of rotation of the wheels and the acceleration will depend upon the angular acceleration of the wheels. Central to balancing our robot is measuring tilt using the IMU, using this feedback to implement closed-loop control. To optimize balance, the centre of mass should be higher relative to the wheel axles since this corresponds to lower angular acceleration and hence slower fall.

2.2 Techniques of Sensor Uses

1. *Accelerometer*: Gives acceleration in multiple axis; x and y calculated using the equation: $\theta = arctan\ (Ay/Ax\ )$. However, noise means this method is likely inaccurate.

2. *Gyrometer*: Calculates the angular velocities across axis. Sometimes has low frequency noise which mean that values are completely wrong.

3. *Gyrometer and Accelerometer*: Data from both are obtained in an MPU6050 and fused using either a Kalman or a Complementary Filter, resulting in an improvement in the results of both sensors.

2.3 DC Motor

DC motors act as continuous actuators to balance the robot, converting electrical energy into mechanical energy through continuous angular rotation to rotate the wheels.

2.4 H-Bridge

A H-Bridge controls the direction of the DC motors by switching the polarity of the voltage applied to the load, allowing the motors to run forward or backward.

The H-bridge has four switches, allowing control of the voltage direction. When switches S1 and S4 are closed and S2 and S3 open, a positive voltage is applied across the motor and vice versa to produce a negative voltage.

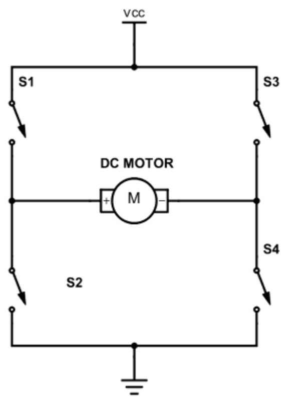


Figure 2 H-Bridge diagram

2.5 PID Control

A PID controller sets the speed of the motors depending on the angle of inclination. It continuously calculates an error value $e(t)$ as the difference between the desired set point and measured process variable and corrects the error based on proportional, integral and derivative terms below:

P: Accounts for present values of the error. For example, if the error is large and positive the control output will also be large and positive.
I: Accounts for past values of the error.
D: Accounts for possible future trends of the error, based on its current rate of change.

The formula using these parameters for PID is below with K_P, K_I, K_D as non-negative parameters.

$$u(t) = K_P e(t) + K_I \int e(t)dt + K_P \frac{de}{dt}$$

3. Components

3.1 List of Components

Table 1

Component	Quantity	Component	Quantity
4TRONIX MOTOR GEARBOX	2	USB extension cable	1
4TRONIX 65mm Yellow wheel	2	Rechargeable battery and charger	1
Arduino Leonardo	1	M3 nuts and bolts	4
GY 521 IMU	1	M4 nuts and bolts	4
SBR4A motor Shield PCB	1	Brackets	2
IMU NANO Shoe PCB	1	Wires	Many
USB cables	2	Plastic plates (1090mm*670mm*30mm)	3
Micro USB adapter	1	Cable ties	1
Switches	2 (Red and Green)		

3.2 Components

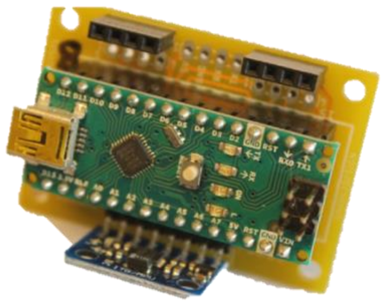


Figure 3 Accelerometer and Gyroscope

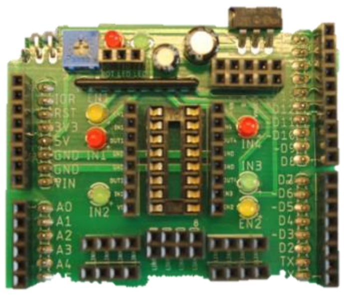


Figure 4 Motor Protoshield

4. Initial Design and Ideas

We initially considered several designs, taking theory into account and focused on creating a feasible yet innovative design.

Our structure was initially divided into 3 layers, with each layer holding a single component, using glue. However, according to the 40 design principles, glue isn't easy to disassemble, making the entire assembly "one-off", which is not ideal in case of any modification, damage or battery replacement. Therefore, we decided to use fastenings such as nuts, bolts and screws instead, so that we did not compromise on a secure fit.

We used aluminium as the main material for our structure due to its availability as well as its material properties such as its toughness, hardness and rigidity, which would minimise rattling and damage.

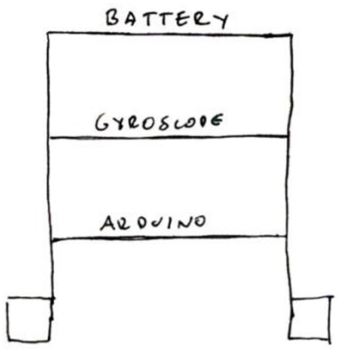


Figure 5 Initial Design

5. Design Improvements and Iterations

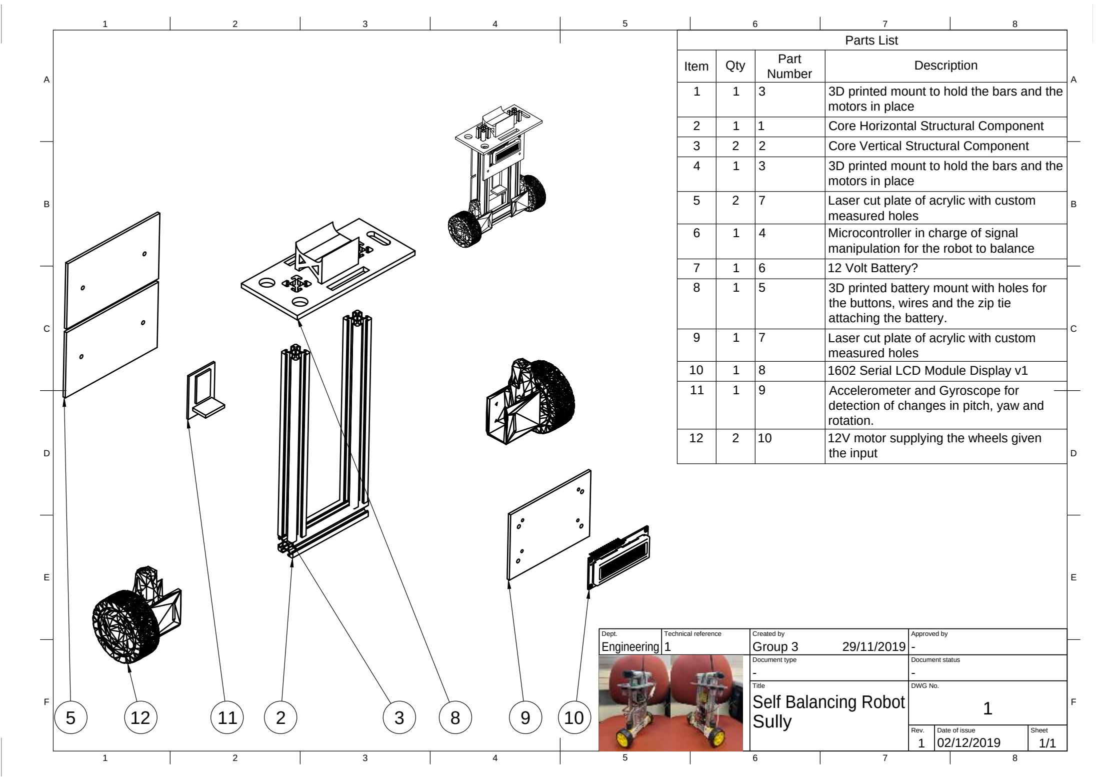
Initially, the robot incorporated two horizontal plates mounted on an aluminium tubed design, due to its equal distribution of mass over the axis of rotation. We learned in order to get more accurate accelerometer readings the accelerometer would have to be mounted vertically. As a result, we were forced to rethink our design.

We realised that it would be more efficient to mount the components vertically rather than horizontally as the robot would be more compact and better shielded against damage as well as provide better accessibility and protection. We also designed and printed a 3D plate to hold the battery over the centre of mass. This first iteration was later adjusted to account for switch holders.

During our testing process, we felt our robot was too unstable to function with our Arduino code. We felt that this may be due to the weight distribution of the robot. Although theory tells us that a robot with a higher centre of mass balances better, we sought to remove any excess space between components and trim the length of the aluminium rods by 4cm. This had the immediate effect of a better performing structure.

6. Assembly Process

6.1 CAD Drawings



Parts List			
Item	Qty	Part Number	Description
1	1	3	3D printed mount to hold the bars and the motors in place
2	1	1	Core Horizontal Structural Component
3	2	2	Core Vertical Structural Component
4	1	3	3D printed mount to hold the bars and the motors in place
5	2	7	Laser cut plate of acrylic with custom measured holes
6	1	4	Microcontroller in charge of signal manipulation for the robot to balance
7	1	6	12 Volt Battery?
8	1	5	3D printed battery mount with holes for the buttons, wires and the zip tie attaching the battery.
9	1	7	Laser cut plate of acrylic with custom measured holes
10	1	8	1602 Serial LCD Module Display v1
11	1	9	Accelerometer and Gyroscope for detection of changes in pitch, yaw and rotation.
12	2	10	12V motor supplying the wheels given the input

Dept. Engineering	Technical reference 1	Created by Group 3 29/11/2019	Approved by -
		Document type -	Document status -
		Title Self Balancing Robot Sully	DWG No. 1
			Rev. 1 · Date of issue 02/12/2019 · Sheet 1/1

Figure 6 Exploded view of robot

6.2 Circuit Design

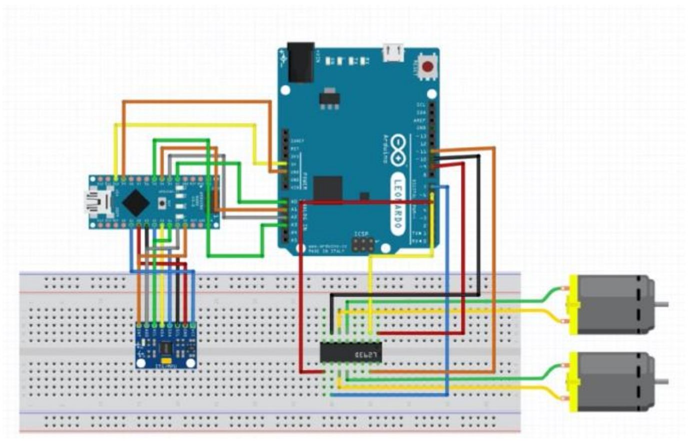


Figure 7 Circuit Diagram

6.3 Assembly Process

The main structure was designed with three core components in mind. The 'U' shape aluminium profile, the 'T shape joints', and the 3D printed brackets. The remaining components should be removable, to allow for easy replacement of components in case of damage, redundancy or improvement.

The U shape is held in place using ‘T-shape joints’ on the rails on the aluminium profiles. Similarly, the acrylic pieces are held together using the rails on the ‘aluminium extrusion bars’ using the ‘t shape joints’ with 4mm screws.

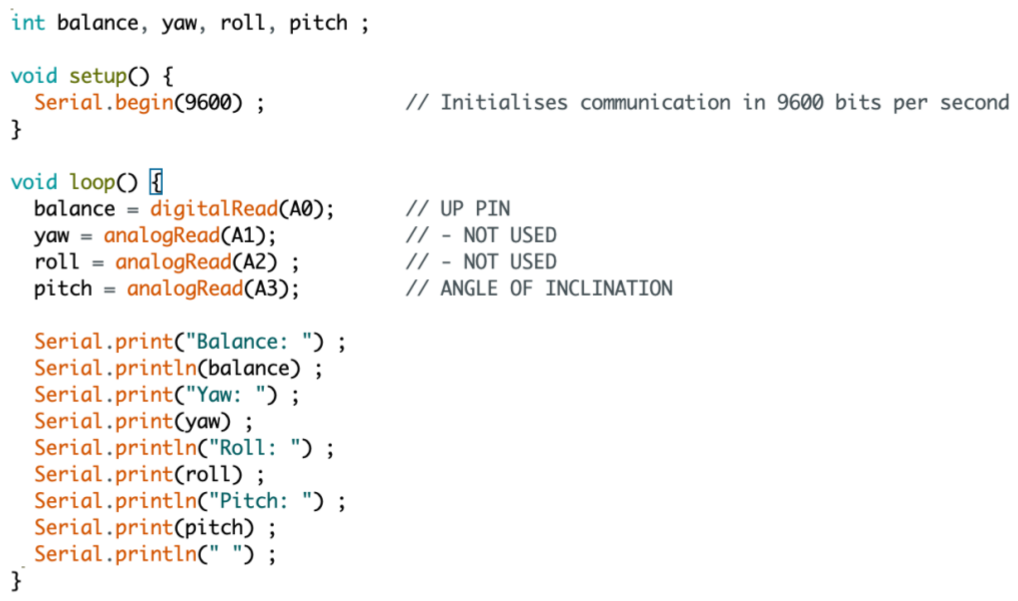
The robot design allows for easy part removal to work on changes and improvements. This was crucial when there were design flaws with the acrylic plates, and we could replace individual plates without disassembling the entire robot. We endeavoured to make the robot as easy to customise as possible. To this effect, we attached the Leonardo plastic motor shield to the structure instead of the Arduino ensuring the Arduino could be easily mounted and unmounted.

7. Code and Testing

7.1 *MPU6050* Testing

The sensor was tested to determine the format of the data and if it functioned as expected. This involves reading values from the GY521 accelerometer and gyroscope. There are 4 different values: yaw, roll, pitch and whether it’s balanced (known as the UP pin)

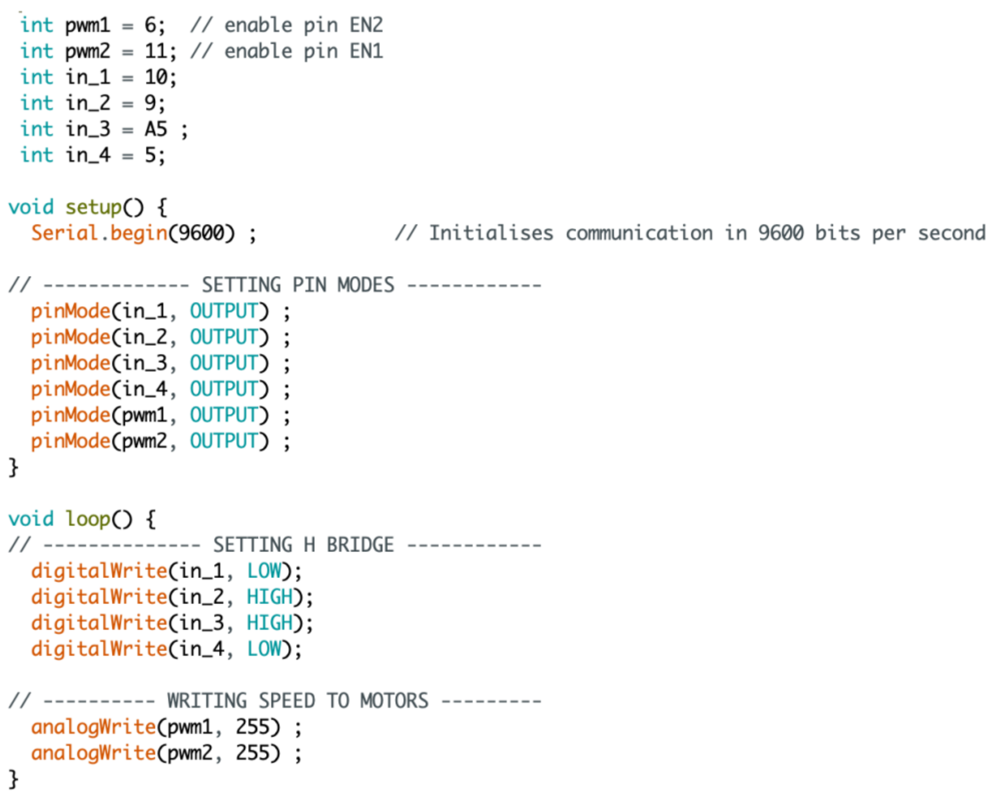
To read data, the following code is used:



```
int balance, yaw, roll, pitch ;

void setup() {
  Serial.begin(9600) ;           // Initialises communication in 9600 bits per second
}

void loop() {
  balance = digitalRead(A0);     // UP PIN
  yaw = analogRead(A1);          // - NOT USED
  roll = analogRead(A2) ;        // - NOT USED
  pitch = analogRead(A3);        // ANGLE OF INCLINATION

  Serial.print("Balance: ") ;
  Serial.println(balance) ;
  Serial.print("Yaw: ") ;
  Serial.print(yaw) ;
  Serial.println("Roll: ") ;
  Serial.print(roll) ;
  Serial.println("Pitch: ") ;
  Serial.print(pitch) ;
  Serial.println(" ") ;
}
```

7.2 DC Motor Testing

Prior to any development of PID, tests were conducted as below to determine whether the wheels functioned as expected. Each motor has 3 pins: two inputs (‘INs’) and an enable (‘EN’) pin.



```
 int pwm1 = 6;  // enable pin EN2
 int pwm2 = 11; // enable pin EN1
 int in_1 = 10;
 int in_2 = 9;
 int in_3 = A5 ;
 int in_4 = 5;

void setup() {
  Serial.begin(9600) ;           // Initialises communication in 9600 bits per second

// ------------- SETTING PIN MODES ------------
  pinMode(in_1, OUTPUT) ;
  pinMode(in_2, OUTPUT) ;
  pinMode(in_3, OUTPUT) ;
  pinMode(in_4, OUTPUT) ;
  pinMode(pwm1, OUTPUT) ;
  pinMode(pwm2, OUTPUT) ;
}

void loop() {
// -------------- SETTING H BRIDGE ------------
  digitalWrite(in_1, LOW);
  digitalWrite(in_2, HIGH);
  digitalWrite(in_3, HIGH);
  digitalWrite(in_4, LOW);

// ---------- WRITING SPEED TO MOTORS ---------
  analogWrite(pwm1, 255) ;
  analogWrite(pwm2, 255) ;
}
```

The test proved successful, as the wheels moved at full speed and changed directions in conjunction with H-bridge initialisation.

7.3 Overall Code Testing

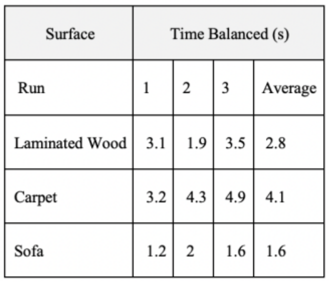
The entire code was enhanced using iterative techniques. To get the robot to balance successfully, values such as K_P, K_I, K_D as well as mapping values were adjusted. The code includes mapping to speed up the motors at small angles and constrain them at mid-high angles, as motors did not adjust well at small angles and to prevent overshoot with higher angles.

8. Experimentation and Testing

8.1 Surface Testing

To determine the best surface on which our robot would succeed, we needed to test on a variety of surfaces:

Table 2

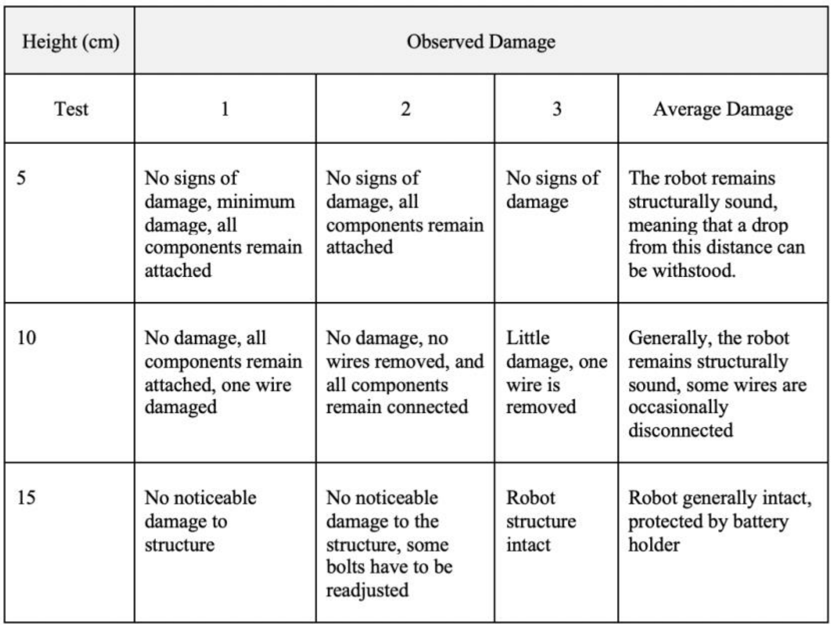


Surface	Time Balanced (s)			
Run	1	2	3	Average
Laminated Wood	3.1	1.9	3.5	2.8
Carpet	3.2	4.3	4.9	4.1
Sofa	1.2	2	1.6	1.6

8.2 Drop and Physical Robustness Testing

To justify the adjustments we made to our design; moving the circuit board to the inside of the robot and creating a 3D printed battery holder that could double as fall protection, we endeavoured to test how well the robot would respond to the falls from different heights. This would simulate how much damage the robot would take in case of failure.

Table 3



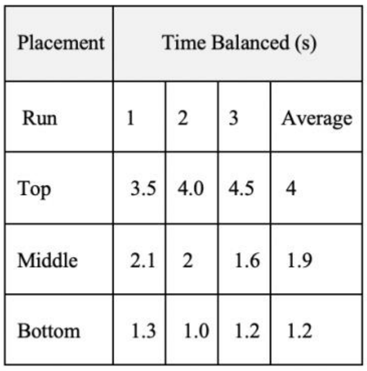
Height (cm)	Observed Damage			
Test	1	2	3	Average Damage
5	No signs of damage, minimum damage, all components remain attached	No signs of damage, all components remain attached	No signs of damage	The robot remains structurally sound, meaning that a drop from this distance can be withstood.
10	No damage, all components remain attached, one wire damaged	No damage, no wires removed, and all components remain connected	Little damage, one wire is removed	Generally, the robot remains structurally sound, some wires are occasionally disconnected
15	No noticeable damage to structure	No noticeable damage to the structure, some bolts have to be readjusted	Robot structure intact	Robot generally intact, protected by battery holder

When dropping from 20cm or higher, cracks began to appear within the acrylic.

8.3 Battery Placement and Alignment with Theory

As mentioned earlier within the theory section, placement of the battery is crucial. This determines how much time we have to drive the motors and hence adjust the robot to stay balanced. We aimed to test if these claims remained true and if, by lowering the placement of our batteries, we could get better results.

Table 4



Placement	Time Balanced (s)			
Run	1	2	3	Average
Top	3.5	4.0	4.5	4
Middle	2.1	2	1.6	1.9
Bottom	1.3	1.0	1.2	1.2

8.4 Overall Testing Results

Our results coincide well with the theory outlined above. When moving the battery, i.e. our centre of mass, the robot performed as expected. Theory states that moving the battery lower would result in a weaker performance, which is what happened. Overall, our robot justified the design decisions made when iterating. This was further reinforced by the drop testing results, which simulated falls we may have should our code fail. Our robot performed best on carpet, as expected, since it provided us with the most grip and traction.

9. Conclusions and Final Thoughts

9.1 Success of the Robot

Overall, we designed a functional robot that used PID control to balance for a maximum of 4.9 seconds. Although we did not meet the objectives fully, we were still able to successfully demonstrate many of the main concepts; electro-mechanical energy conversion, closed loop control and power electronics. We demonstrated electro-mechanical energy conversion through our successful H bridge operation, closed loop control using a gyroscope and the PID code and power electronics through successful operation of our components.

9.2 Potential Improvements to the Process

If we were to complete this task again, we would approach the task in a more organised manner, ensuring that our design was governed more strictly by theoretical knowns, to minimise error. Design wise, we would retrospectively provide the electronic components with more protection by implementing acrylic plates in front of all components, to both protect the robot and improve its aesthetic look. Code and electronics wise we would implement multiple PID controllers. We have currently implemented a single PID controller to control the angle, and by implementing further controls for the speed of the robot, as well as both motors, we feel the robot will perform better.

10. References

Build Electronic Circuits. (2019). *H-bridge-switches - Build Electronic Circuits*. [online] Available at: https://www.build-electronic-circuits.com/h-bridge/h-bridge-switches/ [Accessed 2 Dec. 2019].

Wang, C. (2019). *Professor: Chi-Jo Wang Student's name : Nguyen Van Binh Student ID: MA02B203 Two Wheels Self Balancing Robot 1 Southern Taiwan University Department of. - ppt download*. [online] Slideplayer.com. Available at: https://slideplayer.com/slide/7619597/ [Accessed 1 Dec. 2019].

Tripathy, P. (2019). *Self-Balancing Bot Using Concept of Inverted Pendulum - ethesis*. [online] Ethesis.nitrkl.ac.in. Available at: http://ethesis.nitrkl.ac.in/5262/ [Accessed 2 Dec. 2019].

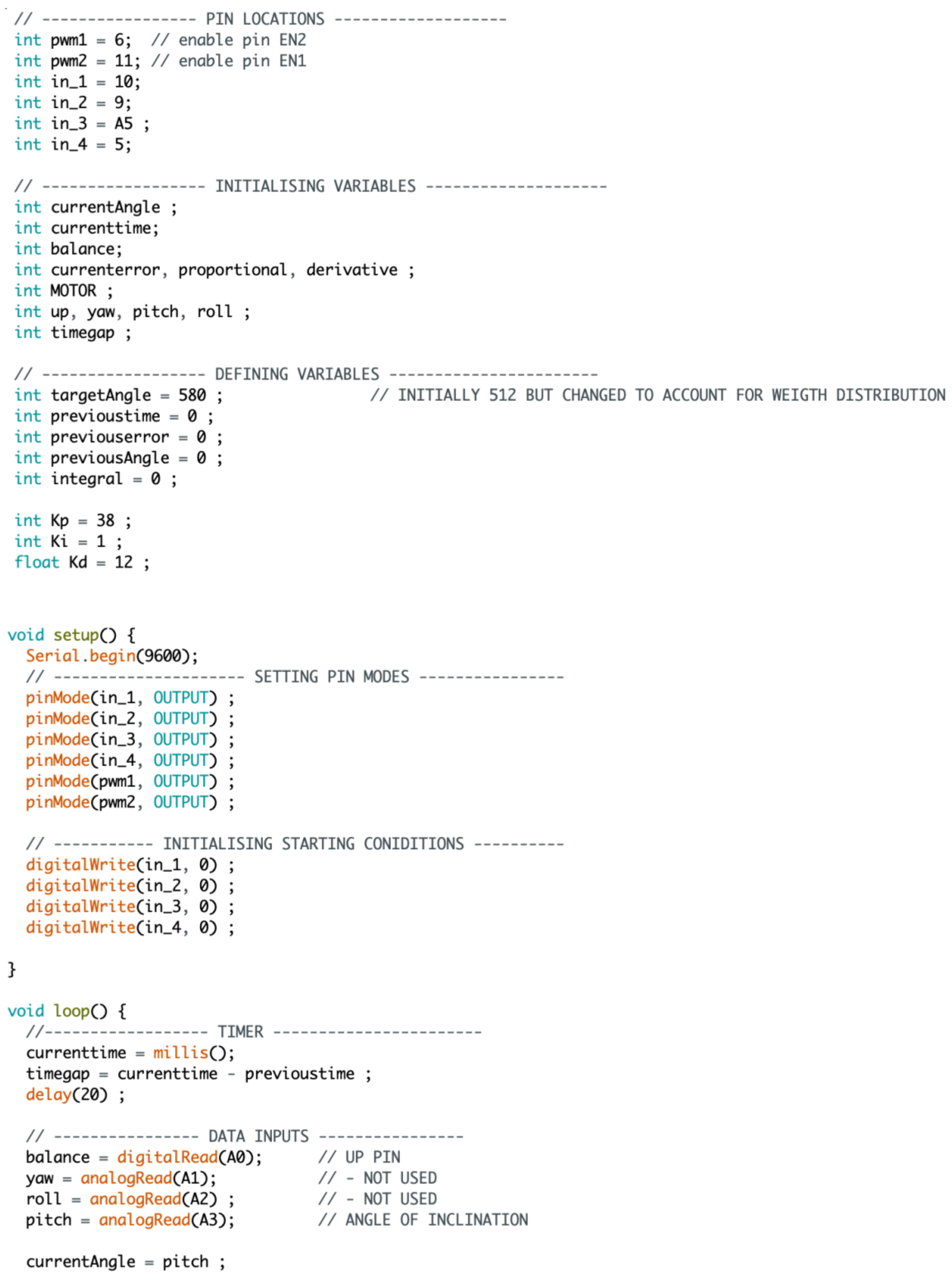
Daware, K. (2019). *How a DC motor works?*. [online] Electricaleasy.com. Available at: https://www.electricaleasy.com/2014/01/basic-working-of-dc-motor.html [Accessed 20 Nov. 2019].

Franz, K. (2019). *What Is an H-Bridge?*. [online] Digilent Inc. Blog. Available at: https://blog.digilentinc.com/what-is-an-h-bridge/ [Accessed 25 Nov. 2019].

M, A. (2019). *PID Theory Explained - National Instruments*. [online] Ni.com. Available at: https://www.ni.com/en-gb/innovations/white-papers/06/pid-theory-explained.html [Accessed 21 Nov. 2019].

11. Appendix

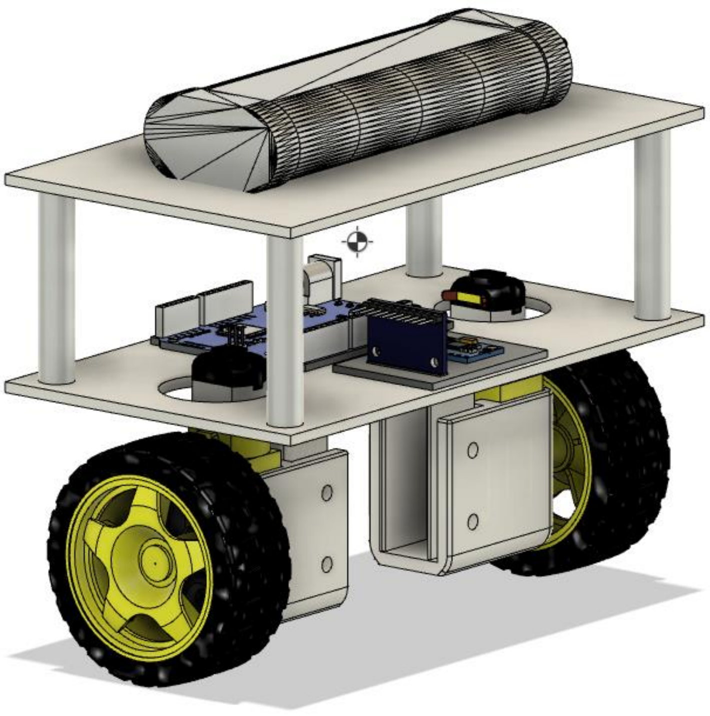
11.1 Overall Code

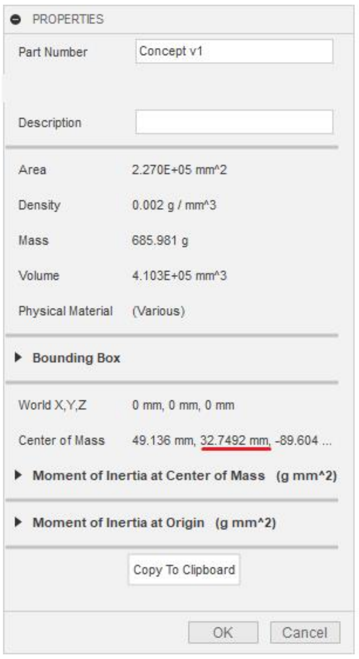


```
// ----------------- PIN LOCATIONS -------------------
int pwm1 = 6;  // enable pin EN2
int pwm2 = 11; // enable pin EN1
int in_1 = 10;
int in_2 = 9;
int in_3 = A5 ;
int in_4 = 5;

// ------------------ INITIALISING VARIABLES --------------------
int currentAngle ;
int currenttime;
int balance;
int currenterror, proportional, derivative ;
int MOTOR ;
int up, yaw, pitch, roll ;
int timegap ;

// ------------------ DEFINING VARIABLES -----------------------
int targetAngle = 580 ;                // INITIALLY 512 BUT CHANGED TO ACCOUNT FOR WEIGTH DISTRIBUTION
int previoustime = 0 ;
int previouserror = 0 ;
int previousAngle = 0 ;
int integral = 0 ;

int Kp = 38 ;
int Ki = 1 ;
float Kd = 12 ;


void setup() {
  Serial.begin(9600);
  // --------------------- SETTING PIN MODES ----------------
  pinMode(in_1, OUTPUT) ;
  pinMode(in_2, OUTPUT) ;
  pinMode(in_3, OUTPUT) ;
  pinMode(in_4, OUTPUT) ;
  pinMode(pwm1, OUTPUT) ;
  pinMode(pwm2, OUTPUT) ;

  // ---------- INITIALISING STARTING CONIDITIONS ----------
  digitalWrite(in_1, 0) ;
  digitalWrite(in_2, 0) ;
  digitalWrite(in_3, 0) ;
  digitalWrite(in_4, 0) ;

}

void loop() {
  //----------------- TIMER ----------------------
  currenttime = millis();
  timegap = currenttime - previoustime ;
  delay(20) ;

  // --------------- DATA INPUTS ----------------
  balance = digitalRead(A0);       // UP PIN
  yaw = analogRead(A1);            // - NOT USED
  roll = analogRead(A2) ;          // - NOT USED
  pitch = analogRead(A3);          // ANGLE OF INCLINATION

  currentAngle = pitch ;
```

```
if (balance == false)     // -------------- IF NOT WITHIN BALANCING RANGE -> ROBOT IS OFF
{
  // SET EVERYTHING TO ZERO
  analogWrite(pwm1, 0) ;
  analogWrite(pwm2, 0) ;
  digitalWrite(in_1, 0);
  digitalWrite(in_2, 0);
  digitalWrite(in_3, 0);
  digitalWrite(in_4, 0);
}

if (balance == true)    // --------------- IF WITHIN BALANCING RANGE -> ROBOT IS ON
{
  //  -------- PID CONTROL ------------
  currenterror = currentAngle - targetAngle ; // works
  proportional = currenterror ;
  integral = integral + (currenterror  * timegap) ;
  integral = constrain(integral, -100, 100) ;
  derivative = (currenterror - previouserror) / timegap ;

  MOTOR = (Kp * proportional) + (Ki * integral) + (Kd * derivative) ;

  MOTOR = map(MOTOR, -5400, 5400, -230, 230) ;
  MOTOR = constrain(MOTOR, -255, 255) ;


  if (MOTOR < 5 && MOTOR > 90)
    {
      MOTOR = map(MOTOR, 5, 90, 60, 120) ;
    }
  else if (MOTOR > -90 && MOTOR < -5)
  {
      MOTOR = map(MOTOR, -90, -5, -120, -60);
  }

 // -------- WRITING TO MOTORS -----------

  if (currenterror < -150)
  {
    // FORWARDS (GLASS SIDE)
    digitalWrite(in_1, HIGH);
    digitalWrite(in_2, LOW);
    digitalWrite(in_3, LOW);
    digitalWrite(in_4, HIGH);
    MOTOR = abs(MOTOR) ;
  }

  else if (currenterror > 150)
  {
    // BACKWARDS (WIRES)
    digitalWrite(in_1, LOW);
    digitalWrite(in_2, HIGH);
    digitalWrite(in_3, HIGH);
    digitalWrite(in_4, LOW);
    MOTOR = abs(MOTOR) ;
  }

  else if (currenterror > -150 && currenterror < -2)
  {
    // FORWARDS (GLASS SIDE)
    digitalWrite(in_1, HIGH);
    digitalWrite(in_2, LOW);
    digitalWrite(in_3, LOW);
    digitalWrite(in_4, HIGH);
    MOTOR = abs(constrain(MOTOR, -180, 180)) ;
  }
I
    else if (currenterror < 150 && currenterror > 2)
    {
      // FORWARDS (GLASS SIDE)
      digitalWrite(in_1, LOW);
      digitalWrite(in_2, HIGH);
      digitalWrite(in_3, HIGH);
      digitalWrite(in_4, LOW);
      MOTOR = abs(constrain(MOTOR, -180, 180)) ;
    }


    else
    {
      MOTOR = 0 ;
    }

    analogWrite(pwm1, MOTOR);
    analogWrite(pwm2, MOTOR);

  }
}
```

11.2 Render of Initial Idea





PROPERTIES

Part Number	Concept v1
Description	
Area	2.270E+05 mm^2
Density	0.002 g / mm^3
Mass	685.981 g
Volume	4.103E+05 mm^3
Physical Material	(Various)

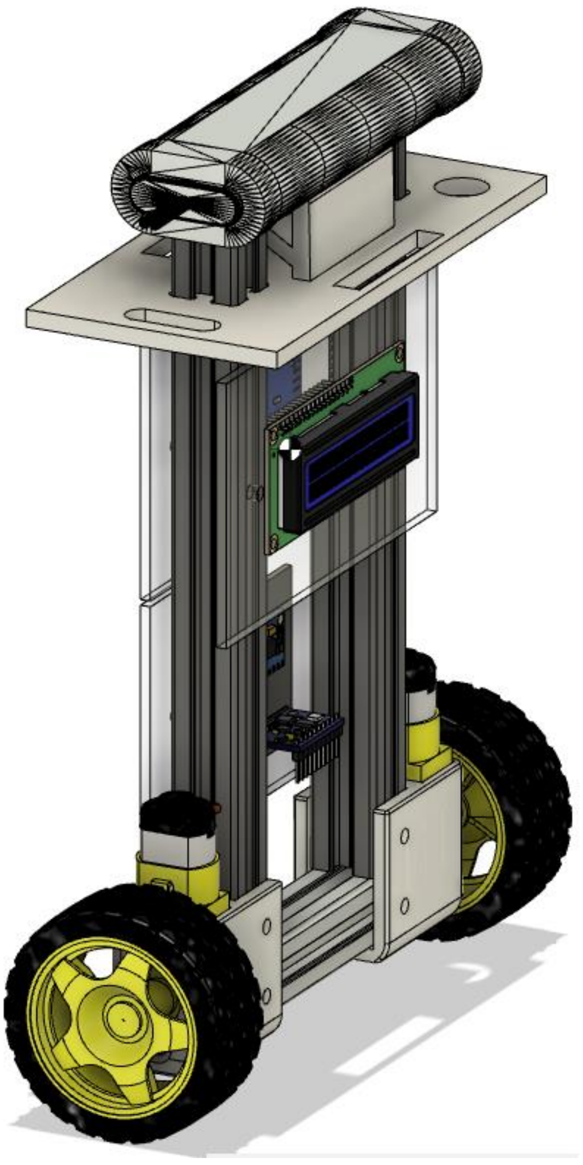
▸ Bounding Box

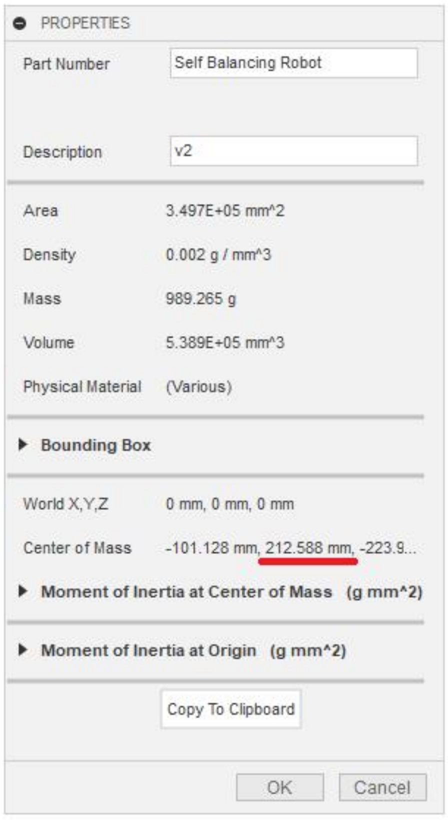
World X,Y,Z	0 mm, 0 mm, 0 mm
Center of Mass	49.136 mm, 32.7492 mm, -89.604 ...

▸ Moment of Inertia at Center of Mass (g mm^2)

▸ Moment of Inertia at Origin (g mm^2)

11.3 Render of Final Idea





PROPERTIES

Part Number	Self Balancing Robot
Description	v2
Area	3.497E+05 mm^2
Density	0.002 g / mm^3
Mass	989.265 g
Volume	5.389E+05 mm^3
Physical Material	(Various)

Bounding Box

World X,Y,Z	0 mm, 0 mm, 0 mm
Center of Mass	-101.128 mm, 212.588 mm, -223.9...

Moment of Inertia at Center of Mass (g mm^2)

Moment of Inertia at Origin (g mm^2)