

BUILDING BALANCING ROBOT

1-Balancing Robot Materials needed

2-Building Balancing Robot Body

3- Wiring

4-Getting Balancing Robot Specific Library

5-Calibrating MPU 6050

6-Changing BT-4 HM10 Bluetooth modül Baut Rate from 115200 to 9600 by using FTD232 Module

7-Downloading Arduino Code

8-Downloading i-phone app for controlling robot (from Apple Store)

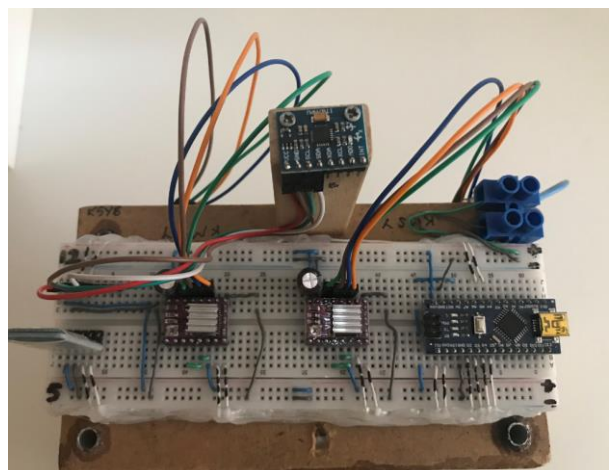
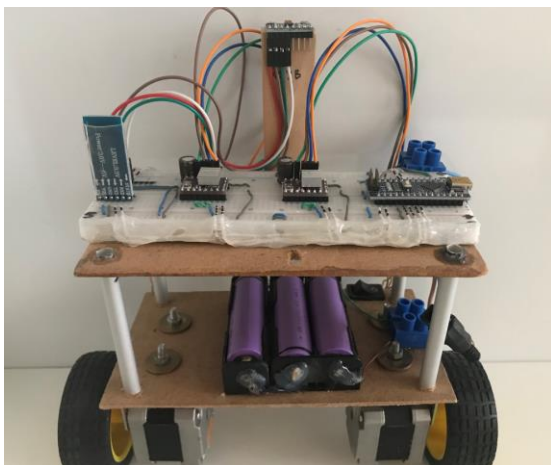
9-Starting the Robot

1-Balancing Robot Materials needed

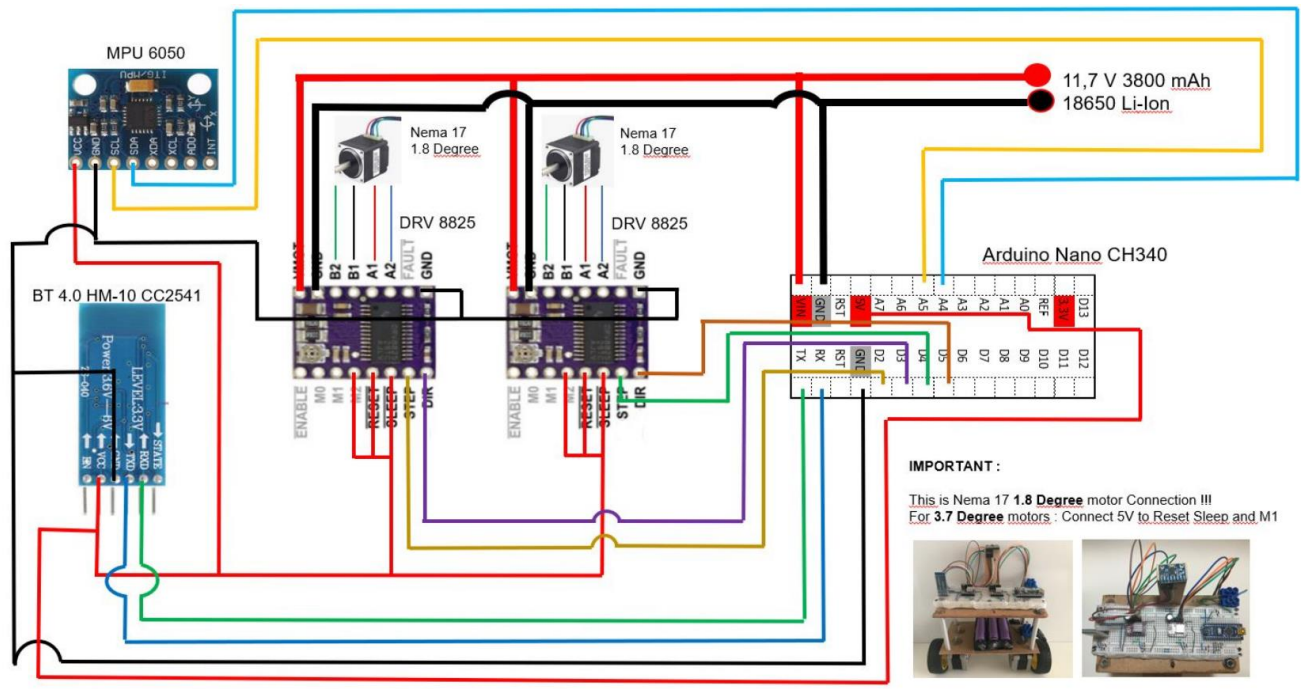
- Arduino Nano V3.0 SMD CH340 Chip
- Bluetooth 4.0 Serial HM-10 CC2541
- MPU6050 3 Axis Gyro
- Nema 17 Step Motors x 2 (1.8 Degree)
- Step Motor Driver DRV8825 x 2
- 3800 mAh, 3.7v Li-ion 18650 Batteries x 3
- Wheels x2
- Materials for Main Body Frame

2-Building Balancing Robot Body

For the wiring, one option is to use Bread Board instead of PCB.



DRV8825 drivers' wiring is very critical. For the Nema Motors with **1.8 Degree step angle**, Just use M2 pin with 5V. Reset and Sleep pins are also 5V connected. For the 3.7 degree Nema motors the connection is different whereas the M1 pin is only activated with 5V in this case !



4- Getting Balancing Robot Specific Library (Get yourself those files from Github by searching)

Go to the Arduino Library directory from Program Files in your PC. Remove all other Libraries in the directory and empty it ! Just copy and paste only the following Libraries. It is really important. Otherwise Arduino gives error when it is run !

- BalanceCar
- Flexitimer2
- I2Cdev
- KalmanFilter
- MPU6050

5-Calibrating MPU 6050

For Calibrating the MPU6050 : Just copy and paste the following code to Arduino. Put the robot in a perfectly horizontal position. Run the code and open Serial Monitor to follow the process. End of the process the MPU is now calibrated.

Many thanks to “Grumpy Old Tech” for all codes below

https://www.youtube.com/channel/UCQp_1InC3IsdWhNjK5xwVg

THE CALIBRATION CODE :

```
// Arduino Calibrate Sketch - used to return offsets for the MPU6050 as each device is slightly different
// - this version: 1.2
// - these code changes by: Grumpy Old Tech - Neville Kripp (12th May 2018)
//
// Changed text output format to be easier to read in the hope less errors will be introduced
// when setting up the MPU.
//
// Feel free to use and abuse my code as you wish taking into account the that my changes are also
// placed under the MIT license.
//

// Arduino sketch that returns calibration offsets for MPU6050
// Version 1.1 (31th January 2014)
// Done by Luis Ródenas <luisrodenaslorda@gmail.com>
// Based on the I2Cdev library and previous work by Jeff Rowberg <jeff@rowberg.net>
// Updates (of the library) should (hopefully) always be available at https://github.com/jrowberg/i2cdevlib

// These offsets were meant to calibrate MPU6050's internal DMP, but can be also useful for reading sensors.
// The effect of temperature has not been taken into account so I can't promise that it will work if you
// calibrate indoors and then use it outdoors. Best is to calibrate and use at the same room temperature.

/* ===== LICENSE =====

I2Cdev device library code is placed under the MIT license

Copyright (c) 2011 Jeff Rowberg

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
```

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.

=====

*/

// I2Cdev and MPU6050 must be installed as libraries

#include "I2Cdev.h"

#include "MPU6050.h"

#include "Wire.h"

////////////////////// CONFIGURATION //////////////////////////////////////

//Change this 3 variables if you want to fine tune the sketch to your needs.

int buffersize=1000; //Amount of readings used to average, make it higher to get more precision but sketch will be slower (default:1000)

int accel_deadzone=8; //Accelerometer error allowed, make it lower to get more precision, but sketch may not converge (default:8)

int giro_deadzone=1; //Giro error allowed, make it lower to get more precision, but sketch may not converge (default:1)

// default I2C address is 0x68

// specific I2C addresses may be passed as a parameter here

// AD0 low = 0x68 (default for InvenSense evaluation board)

// AD0 high = 0x69

//MPU6050 accelgyro;

MPU6050 accelgyro(0x68); // <-- use for AD0 high

int16_t ax, ay, az, gx, gy, gz;

int mean_ax, mean_ay, mean_az, mean_gx, mean_gy, mean_gz, state=0;

int ax_offset, ay_offset, az_offset, gx_offset, gy_offset, gz_offset;

////////////////////// SETUP //////////////////////////////////////

void setup() {

```

// join I2C bus (I2Cdev library doesn't do this automatically)

Wire.begin();

// COMMENT NEXT LINE IF YOU ARE USING ARDUINO DUE

TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz). Leonardo measured 250kHz.


// initialize serial communication
Serial.begin(9600);


// initialize device
accelgyro.initialize();


// wait for ready
while (Serial.available() && Serial.read()); // empty buffer
while (!Serial.available()){
  Serial.println(F("Send any character to start sketch.\n"));
  delay(1500);
}
while (Serial.available() && Serial.read()); // empty buffer again


// start message
Serial.println("\nMPU6050 Calibration Sketch");

delay(2000);

Serial.println("\nYour MPU6050 should be placed in horizontal position, with package letters facing up. \nDon't touch it until you see a
finish message.\n");

delay(3000);

// verify connection
Serial.println(accelgyro.testConnection() ? "MPU6050 connection successful" : "MPU6050 connection failed");

delay(1000);

// reset offsets
accelgyro.setXAccelOffset(0);
accelgyro.setYAccelOffset(0);
accelgyro.setZAccelOffset(0);
accelgyro.setXGyroOffset(0);
accelgyro.setYGyroOffset(0);
accelgyro.setZGyroOffset(0);
}

//////////////////// LOOP //////////////////////////////////////

```

```

void loop() {

  if (state==0){

    Serial.println("\nReading sensors for first time...");

    meansensors();

    state++;

    delay(1000);

  }


  if (state==1) {

    Serial.println("\nCalculating offsets...");

    calibration();

    state++;

    delay(1000);

  }


  if (state==2) {

    meansensors();

    Serial.println("\nFINISHED!");

    Serial.print("\nSensor readings with offsets:\taceIX: ");

    Serial.print(mean_ax);

    Serial.print("\taceIY: ");

    Serial.print(mean_ay);

    Serial.print("\taceIZ: ");

    Serial.print(mean_az);

    Serial.print("\tgiroX: ");

    Serial.print(mean_gx);

    Serial.print("\tgiroY: ");

    Serial.print(mean_gy);

    Serial.print("\tgiroZ: ");

    Serial.println(mean_gz);

    Serial.print("          Your offsets:\taceIX: ");

    Serial.print(ax_offset);

    Serial.print("\taceIY: ");

    Serial.print(ay_offset);

    Serial.print("\taceIZ: ");

    Serial.print(az_offset);

    Serial.print("\tgiroX: ");

    Serial.print(gx_offset);

```

```

Serial.print("\tgiroY: ");

Serial.print(gy_offset);

Serial.print("\tgiroZ: ");

Serial.println(gz_offset);


Serial.println("\n\nCheck that your sensor readings are close to 0 0 16384 0 0 0");

Serial.println("If calibration was succesful write down your offsets so you can set them in your projects using something similar to
mpu.setXAccelOffset(youroffset)");

while (1);

}

}

//////////////////////////////////// FUNCTIONS //////////////////////////////////////

void meansensors(){

long i=0,buff_ax=0,buff_ay=0,buff_az=0,buff_gx=0,buff_gy=0,buff_gz=0;


while (i<(bufferSize+101)){

// read raw accel/gyro measurements from device

accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);


if (i>100 && i<=(bufferSize+100)){ //First 100 measures are discarded

buff_ax=buff_ax+ax;

buff_ay=buff_ay+ay;

buff_az=buff_az+az;

buff_gx=buff_gx+gx;

buff_gy=buff_gy+gy;

buff_gz=buff_gz+gz;

}

if (i==(bufferSize+100)){

mean_ax=buff_ax/bufferSize;

mean_ay=buff_ay/bufferSize;

mean_az=buff_az/bufferSize;

mean_gx=buff_gx/bufferSize;

mean_gy=buff_gy/bufferSize;

mean_gz=buff_gz/bufferSize;

}

i++;

delay(2); //Needed so we don't get repeated measures

```

```
}  
}
```

```
void calibration(){  
  
  ax_offset=-mean_ax/8;  
  
  ay_offset=-mean_ay/8;  
  
  az_offset=(16384-mean_az)/8;  
  
  
  gx_offset=-mean_gx/4;  
  
  gy_offset=-mean_gy/4;  
  
  gz_offset=-mean_gz/4;  
  
  while (1){  
  
    int ready=0;  
  
    accelgyro.setXAccelOffset(ax_offset);  
  
    accelgyro.setYAccelOffset(ay_offset);  
  
    accelgyro.setZAccelOffset(az_offset);  
  
  
    accelgyro.setXGyroOffset(gx_offset);  
    accelgyro.setYGyroOffset(gy_offset);  
    accelgyro.setZGyroOffset(gz_offset);  
  
  
    meansensors();  
  
    Serial.println("...");  
  
  
    if (abs(mean_ax)<=acel_deadzone) ready++;  
    else ax_offset=ax_offset-mean_ax/acel_deadzone;  
  
  
    if (abs(mean_ay)<=acel_deadzone) ready++;  
    else ay_offset=ay_offset-mean_ay/acel_deadzone;  
  
  
    if (abs(16384-mean_az)<=acel_deadzone) ready++;  
    else az_offset=az_offset+(16384-mean_az)/acel_deadzone;  
  
  
    if (abs(mean_gx)<=giro_deadzone) ready++;  
    else gx_offset=gx_offset-mean_gx/(giro_deadzone+1);  
  
  
    if (abs(mean_gy)<=giro_deadzone) ready++;  
    else gy_offset=gy_offset-mean_gy/(giro_deadzone+1);  
  
  }  
}
```



```

if (abs(mean_gz)<=giro_deadzone) ready++;
else gz_offset=gz_offset-mean_gz/(giro_deadzone+1);

if (ready==6) break;
}
}

```

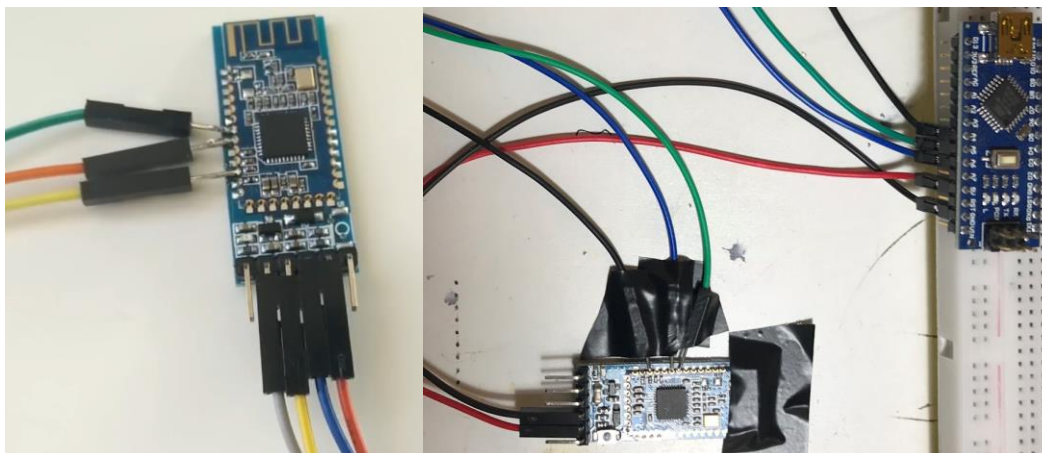
6-Changing BT-4 HM10 Bluetooth module Baut Rate from 115200 to 9600 by using FTD232 Module

HM-10 is coming normally with 115200 baut rate.To be able to use i-phone for controlling the robot, the baut rate should be 9600 ! For this change, you should install AT Software on your HM-10.

STEP-1

The HM-10 and Arduino Nano connections are made as follows:

Cables will be soldered to pins 7, 8, 11 on the HM-10. (Numbering is from top to bottom). 3 cables coming from HM-10 will be connected on Arduino as 7-> A5, 8-> A6, 11-> A4.



STEP-2

On the PC, **CCLoader.exe** program will be run. The program is in the Github as in following address. Many Thanks to “DevelopandPlay” for the useful video and all the softwares.

<https://www.youtube.com/watch?v=e3491-v8Og&t=6s>

<https://github.com/RedBearLab/CCLoader/tree/master/Windows>

To run CCLoader.exe, Go to the Command System and find CCLoader in the appropriate folder and Run. After running the program, then the following line is shown on the PC screen as per below

```
C:\> Eingabeaufforderung

Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Martin>cd /d C:/Users/Martin/Downloads

C:\Users\Martin\Downloads>CCLoader.exe 5 CC2541hm10v540.bin
Copyright (c) 2013 RedBearLab.com
CCLoader.exe version 0.5
Invalid parameters.
Usage: CCLoader.exe <com number> <bin file> <device>
Example: CCLoader.exe 2 abc.bin 0
<device>: 0 -- Default (e.g. UNO)
          1 -- Leonardo

C:\Users\Martin\Downloads>CCLoader.exe 5 CC2541hm10v540.bin 0_
```

Enter 0 (Zero) end of line and Enter (In here 4 is Port Number !!! 0 is Arduino Nano). When Enter it counts from 1 to 500 and completes the process. At end of counting AT Software has been uploaded to BT.

```
C:\> Eingabeaufforderung - CCLoader.exe 5 CC2541hm10v540.bi...

* If there is no respond last for 3s, please press "Ctrl+C" to exit!
* And pay attention to :
*   1. The connection between computer and Arduino;
*   2. The connection between Arduino and CC2540;
*   3. Whether the device you using is Leonardo or not;
*   4. Other unexpected errors.
/...../

Waiting for respond from Arduino...

Uploading firmware...

1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
3  24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
3  44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
3  64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84
3  84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103
103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119
119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135
135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151
151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167
167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183
183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199
199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
215 216 217 218 219 220 221
```

STEP-3

Uploading Code to FT232 Module :

Software serialexample.ino software will be uploaded. The ino code is as per below. Just Copy and Paste it to Arduino...(Be careful that Software serial.h is in the Arduino library)

Serial example Code :

```
#include <SoftwareSerial.h>
```

```
SoftwareSerial BTSerial(3,4); //RX|TX (ÖNCE BT KART TAKILI İKEN PROGRAMI UPLOAD ET. SONRA BT kartı takıp çıkart ve AT komutu gönder!!!!)
```

```
void setup(){  
  Serial.begin(9600);  
  BTSerial.begin(9600); // default baud rate  
  while(!Serial); //if it is an Arduino Micro  
  Serial.println("AT commands: ");  
}  
  
void loop(){  
  //read from the HM-10 and print in the Serial  
  if(BTSerial.available())  
    Serial.write(BTSerial.read());  
  
  //read from the Serial and print to the HM-10  
  if(Serial.available())  
    BTSerial.write(Serial.read());  
}
```

After uplodng the software then Serial monitor will turn on. When AT is typed on the Serial Monitor line and entered, it will be seen that it says “OK” on the screen. It means that the communication has been succesfully established. Now It is very easy now to change the Baut Rate with simple AT commands !

Just Send AT+BAUD4 command by typing in the serial monitor and Enter, then it will set the baud rate to 9600 !!!

7-Downloading Arduino Codes :

In the Arduino Code Directory on the PC, the following files should be located.

MPU.cpp

MPU.h

PID.cpp

PID.h

SkaterBot.ino

The visual of the Arduino Code screen should be as per below.



```
SkaterBot | Arduino 1.8.12
Dosya Düzenle Taslak Araçlar Yardım

SkaterBot MPU.cpp MPU.h PID.cpp PID.h

}
else if (throttleCounterLeftMotor == 1) {
    PORTD |= 0b00000100; //Set output 2 high to create a pulse for the stepper controller
}
else if (throttleCounterLeftMotor == 2) {
    PORTD &= 0b11111011; //Set output 2 low because the pulse only has to last for 20us
}

// Right motor pulse calculations
throttleCounterRightMotor++; //Increase the throttle_counter_right_motor variable by 1 every time the routine is executed
if (throttleCounterRightMotor > throttleRightMotorMemory) { //If the number of loops is larger then the throttle_right_motor_memory variable
    throttleCounterRightMotor = 0; //Reset the throttle_counter_right_motor variable
    throttleRightMotorMemory = throttleRightMotor; //Load the next throttle_right_motor variable
    if (throttleRightMotorMemory < 0) { //If the throttle_right_motor_memory is negative
        PORTD |= 0b00100000; //Set output 5 low to reverse the direction of the stepper controller
        throttleRightMotorMemory *= -1; //Invert the throttle_right_motor_memory variable
    }
    else {
        PORTD &= 0b11011111; //Set output 5 high for a forward direction of the stepper motor
    }
}
else if (throttleCounterRightMotor == 1) {
    PORTD |= 0b00010000; //Set output 4 high to create a pulse for the stepper controller
}
else if (throttleCounterRightMotor == 2) {
    PORTD &= 0b11101111; //Set output 4 low because the pulse only has to last for 20us
}
}
```

Arduino Main Code :

Just Copy and Paste the following Lines to Arduino

```
//
// Terms of use
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
```

```
// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
// THE SOFTWARE.
```

```
//
```

```
// SkaterBot
```

```
//
```

```
#include "MPU.h"
```

```
#include "PID.h"
```

```
// Pins
```

```
#define batteryLowIndicator 13
```

```
#define leftPulse    2
```

```
#define leftDirection  3
```

```
#define rightPulse    4
```

```
#define rightDirection 5
```

```
// PID Paramaters
```

```
#define pidP  15
```

```
#define pidI  1.5
```

```
#define pidD  2
```

```
#define pidDB  5
```

```
#define pidOPMin -400
```

```
#define pidOPMax 400
```

```
// Dynamic Paramaters
```

```
float turnSpeed = 30;
```

```
float maxSpeed = 150;
```

```
// State Engine
```

```
int robotState;
```

```
#define Starting 0
```

```
#define Balancing 1
```

```
// Battery Monitoring
```

```
int  batteryVoltage;
```

```
bool lowBattery;
```

```

// Bluetooth Comms

byte receivedByte;

int receivedCounter;


// Loop Timing

unsigned long loopTimer;

unsigned long reportCounter;


// Control Variables

float angle;

float outputLeft;

float outputRight;


int leftMotor;

int throttleLeftMotor;

int throttleCounterLeftMotor;

int throttleLeftMotorMemory;


int rightMotor;

int throttleRightMotor;

int throttleCounterRightMotor;

int throttleRightMotorMemory;


// Class Instances

MPU mpu(0x68);    // MPU-6050 I2C address (0x68 or 0x69)

PID pid;


//*****

// SETUP

//*****


void setup() {

    Serial.begin(9600);    //Start the serial port at 9600 bps


    pinMode(batteryLowIndicator, OUTPUT);

    pinMode(leftPulse, OUTPUT);

```

```

pinMode(leftDirection, OUTPUT);

pinMode(rightPulse, OUTPUT);

pinMode(rightDirection, OUTPUT);


//To create a variable pulse for controlling the stepper motors a timer is created that will execute a piece of code (subroutine) every 20us
//This subroutine is called TIMER2_COMPA_vect

TCCR2A = 0; //Make sure that the TCCR2A register is set to zero
TCCR2B = 0; //Make sure that the TCCR2A register is set to zero
TIMSK2 |= (1 << OCIE2A); //Set the interrupt enable bit OCIE2A in the TIMSK2 register
TCCR2B |= (1 << CS21); //Set the CS21 bit in the TCCRB register to set the prescaler to 8
OCR2A = 39; //The compare register is set to 39 => 20us / (1s / (16.000.000MHz / 8)) - 1
TCCR2A |= (1 << WGM21); //Set counter 2 to CTC (clear timer on compare) mode


// Setup PID
pid.reset();
pid.setTuningParameters(pidP, pidI, pidD);
pid.setDeadband(pidDB);
pid.setOutputLimited(pidOPMin, pidOPMax);


// Initialise the MPU
mpu.initialise();


// get calibration values
mpu.getGyroCalibrationValues();


robotState = Starting;
loopTimer = micros() + 4000;
}


//*****

// LOOP

//*****


void loop() {

if (Serial.available()) {
receivedByte = Serial.read();

```

```

    receivedCounter = 0;
}

if (receivedCounter <= 25) {
    receivedCounter++;
}
else {
    receivedByte = 0x00;
}

batteryVoltage = map(analogRead(0),0,1023,0,1250);

if (batteryVoltage < 1070 && batteryVoltage > 1050) {
    Serial.print("Battery Warning");
}

if (batteryVoltage < 1050 && batteryVoltage > 800) {
    digitalWrite(batteryLowIndicator, HIGH);
    lowBattery = true;
    Serial.print("Battery Low");
}

// State Machine
switch (robotState) {
    case Starting:
    {
        angle = mpu.getAccelAngle();
        if (angle > -0.5 && angle < 0.5) {
            mpu.setTiltAngle(angle);
            robotState = Balancing;
        }
        break;
    }
    case Balancing:
    {
        angle = mpu.getTiltAngle();
        if (angle > 30 || angle < -30 || lowBattery) {

```



```

    robotState = Starting;

    pid.reset();
}

else {

    // Process PID

    pid.processVariable = angle;

    pid.calculate();

}

break;

}

}

// Calculate control signals

outputLeft = pid.output;

outputRight = pid.output;

// Turn Left

if (receivedByte & 0b00000010) {

    outputLeft += turnSpeed;

    outputRight -= turnSpeed;

}

// Turn Right

if (receivedByte & 0b00000001) {

    outputLeft -= turnSpeed;

    outputRight += turnSpeed;

}

// Forward

if (receivedByte & 0b00000100) {

    if (pid.setpoint > -2.5) {

        pid.setpoint -= 0.05;

    }

    if (pid.output > -maxSpeed) {

        pid.setpoint -= 0.005;

    }

}

```

```

// Reverse

if (receivedByte & 0b00001000) {
    if (pid.setpoint < 2.5) {
        pid.setpoint += 0.05;
    }
    if (pid.output < maxSpeed) {
        pid.setpoint += 0.005;
    }
}

// Not forward or reverse
if (!(receivedByte & 0b00001100)) {
    if (pid.setpoint > 0.5) {
        pid.setpoint -= 0.05;
    }
    else if (pid.setpoint < -0.5) {
        pid.setpoint += 0.05;
    }
    else {
        pid.setpoint = 0;
    }
}

if (pid.setpoint == 0) {
    if (pid.output < 0) {
        pid.selfBalanceSetpoint += 0.0015;
    }
    if (pid.output > 0) {
        pid.selfBalanceSetpoint -= 0.0015;
    }
}

// Motor Calculations

// Compensate for the non-linear behaviour of the stepper motors
if (outputLeft > 0) {
    outputLeft = 405 - (1/(outputLeft + 9)) * 5500;
}

```

```

else if (outputLeft < 0) {
    outputLeft = -405 - (1/(outputLeft - 9)) * 5500;
}

if (outputRight > 0) {
    outputRight = 405 - (1/(outputRight + 9)) * 5500;
}
else if (outputRight < 0) {
    outputRight = -405 - (1/(outputRight - 9)) * 5500;
}

// Calculate the needed pulse time for the left and right stepper motor controllers
if (outputLeft > 0) {
    leftMotor = 400 - outputLeft;
}
else if (outputLeft < 0) {
    leftMotor = -400 - outputLeft;
}
else {
    leftMotor = 0;
}

if (outputRight > 0) {
    rightMotor = 400 - outputRight;
}
else if (outputRight < 0) {
    rightMotor = -400 - outputRight;
}
else {
    rightMotor = 0;
}

// Copy for interrupt to use
throttleLeftMotor = leftMotor;
throttleRightMotor = rightMotor;

if (reportCounter > 50) {
    reportCounter = 0;
}

```

```

// Serial.print(angle);
// Serial.print(",");
// Serial.print(pid.selfBalanceSetpoint);
// Serial.print(",");
// Serial.print(pid.output);
// Serial.print(",");
// Serial.print(throttleLeftMotor);
// Serial.print(",");
// Serial.print(throttleRightMotor);
// Serial.println("");
}

// Delay 4 milliseconds
while(loopTimer > micros());
loopTimer += 4000;
reportCounter ++;
}

//*****
// INTERRUPT HANDLERS
//*****

ISR(TIMER2_COMPA_vect) {

// Left motor pulse calculations

throttleCounterLeftMotor ++;           //Increase the throttle_counter_left_motor variable by 1 every time this routine is
executed

if (throttleCounterLeftMotor > throttleLeftMotorMemory) { //If the number of loops is larger then the throttle_left_motor_memory
variable

throttleCounterLeftMotor = 0;           //Reset the throttle_counter_left_motor variable

throttleLeftMotorMemory = throttleLeftMotor; //Load the next throttle_left_motor variable

if (throttleLeftMotorMemory < 0) {       //If the throttle_left_motor_memory is negative

PORTD &= 0b11110111;                   //Set output 3 low to reverse the direction of the stepper controller

throttleLeftMotorMemory *= -1;          //Invert the throttle_left_motor_memory variable

}

else {

PORTD |= 0b00001000;                   // Set output 3 high for a forward direction of the stepper motor

}
}
}

```

```

}

else if (throttleCounterLeftMotor == 1) {

    PORTD |= 0b00000100;          //Set output 2 high to create a pulse for the stepper controller

}

else if (throttleCounterLeftMotor == 2) {

    PORTD &= 0b11111011;          //Set output 2 low because the pulse only has to last for 20us

}


// Right motor pulse calculations

throttleCounterRightMotor++;      //Increase the throttle_counter_right_motor variable by 1 every time the routine is
executed

if (throttleCounterRightMotor > throttleRightMotorMemory) { //If the number of loops is larger then the throttle_right_motor_memory
variable

    throttleCounterRightMotor = 0;      //Reset the throttle_counter_right_motor variable

    throttleRightMotorMemory = throttleRightMotor;    //Load the next throttle_right_motor variable

    if (throttleRightMotorMemory < 0) {      //If the throttle_right_motor_memory is negative

        PORTD |= 0b00100000;          //Set output 5 low to reverse the direction of the stepper controller

        throttleRightMotorMemory *= -1;      //Invert the throttle_right_motor_memory variable

    }

    else {

        PORTD &= 0b11011111;          //Set output 5 high for a forward direction of the stepper motor

    }

}

else if (throttleCounterRightMotor == 1) {

    PORTD |= 0b00010000;          //Set output 4 high to create a pulse for the stepper controller

}

else if (throttleCounterRightMotor == 2) {

    PORTD &= 0b11101111;          //Set output 4 low because the pulse only has to last for 20us

}

}

```

MPU.ccp Code :

Create a file in the Arduino named MPU.ccp and copy and paste the following lines

```

//
// MPU Class
//

```

```

#include "Arduino.h"

#include "MPU.h"

MPU::MPU(int address) {

    busAddress = address;

    Wire.begin(); //Start the I2C bus as master
    TWBR = 12;    //Set the I2C clock speed to 400kHz
}

void MPU::initialise() {

    //reset();
    wakeup();
    setGyroFullScaleRange();
    setAccelFullScaleRange();
    setLowPassFilter();
}

void MPU::getGyroCalibrationValues() {

    int loopCounter;

    for (loopCounter = 0; loopCounter < 500; loopCounter++) {

        gyroYawCalibrationValue += getRotationX();
        gyroPitchCalibrationValue += getRotationY();
        delayMicroseconds(4000);
    }

    gyroYawCalibrationValue /= 500;
    gyroPitchCalibrationValue /= 500;
}

float MPU::getAccelAngle() {

    long accelRawData;

```

```
float accelAngle;
```

```
//Get the Accel Angle
```

```
accelRawData = getAccelerationZ() + accelCalibrationValue;
```

```
constrain(accelRawData, -8200, 8200);
```

```
accelAngle = asin((float)accelRawData / 8200.0) * RAD2DEGREE;
```

```
return accelAngle;
```

```
}
```

```
void MPU::setTiltAngle(float angle) {
```

```
    tiltAngle = angle;
```

```
}
```

```
float MPU::getTiltAngle() {
```

```
    long gyroYawRawData;
```

```
    long gyroPitchRawData;
```

```
    long accelRawData;
```

```
    float accelAngle;
```

```
//Get the Accel Angle
```

```
accelRawData = getAccelerationZ() + accelCalibrationValue;
```

```
constrain(accelRawData, -8200, 8200);
```

```
accelAngle = asin((float)accelRawData / 8200.0) * RAD2DEGREE;
```

```
gyroYawRawData = getRotationX();
```

```
gyroPitchRawData = getRotationY() - gyroPitchCalibrationValue;
```

```
tiltAngle += gyroPitchRawData * -0.000031; // update with this loops travelled value // orijinal deđer 0.000031 // 2. Robot -0.000031 ile ÇALIŞTI
```

```
// Compensate for mounting try -0.0000003 or 0.0000003 if required
```

```
gyroYawRawData -= gyroYawCalibrationValue;
```

```
//tiltAngle -= gyroYawRawData * 0.0000003;
```

```
// Correct the drift of the gyro angle with the accelerometer angle
```

```
tiltAngle = tiltAngle * 0.9996 + accelAngle * 0.0004;
```

```
return tiltAngle;
```

```
}
```

```
void MPU::reset() {
```

```
    Wire.beginTransaction(busAddress);  
    Wire.write(MPU_REG_PWR_MGMT_1);  
    Wire.write(0b10000000);  
    Wire.endTransmission();  
}
```

```
void MPU::wakeup() {
```

```
    Wire.beginTransaction(busAddress);  
    Wire.write(MPU_REG_PWR_MGMT_1);  
    Wire.write(0b00000000);  
    Wire.endTransmission();  
}
```

```
void MPU::setGyroFullScaleRange() {
```

```
    Wire.beginTransaction(busAddress);  
    Wire.write(MPU_REG_GYRO_CONFIG);  
    Wire.write(0b00000000);    // +/- 250 derees per second  
    Wire.endTransmission();  
}
```

```
void MPU::setAccelFullScaleRange() {
```

```
    Wire.beginTransaction(busAddress);  
    Wire.write(MPU_REG_ACCEL_CONFIG);  
    Wire.write(0b00001000);    // +/- 4G  
    Wire.endTransmission();  
}
```

```
void MPU::setLowPassFilter() {
```

```
    Wire.beginTransaction(busAddress);  
    Wire.write(MPU_REG_CONFIG);
```



```
Wire.write(0b00000011);    // 42/44 Hz
Wire.endTransmission();
}
```

```
long MPU::getRotationX() {

    long value;

    Wire.beginTransmission(busAddress);
    Wire.write(MPU_REG_GYRO_XOUT_H);
    Wire.endTransmission();

    Wire.requestFrom(busAddress, 2);

    value = Wire.read() << 8 | Wire.read();

    return value;
}
```

```
long MPU::getRotationY() {

    long value;

    Wire.beginTransmission(busAddress);
    Wire.write(MPU_REG_GYRO_YOUT_H);
    Wire.endTransmission();

    Wire.requestFrom(busAddress, 2);

    value = Wire.read() << 8 | Wire.read();

    return value;
}
```

```
long MPU::getRotationZ() {

    long value;

    Wire.beginTransmission(busAddress);
    Wire.write(MPU_REG_GYRO_ZOUT_H);
    Wire.endTransmission();

    Wire.requestFrom(busAddress, 2);

    value = Wire.read() << 8 | Wire.read();

    return value;
}
```

```
long MPU::getAccelerationX() {
```

```

long value;

Wire.beginTransmission(busAddress);

Wire.write(MPU_REG_ACCEL_XOUT_H);

Wire.endTransmission();

Wire.requestFrom(busAddress, 2);

value = Wire.read() << 8 | Wire.read();

return value;

}

```

```

long MPU::getAccelerationY() {

```

```

    long value;

    Wire.beginTransmission(busAddress);

    Wire.write(MPU_REG_ACCEL_YOUT_H);

    Wire.endTransmission();

    Wire.requestFrom(busAddress, 2);

    value = Wire.read() << 8 | Wire.read();

    return value;

}

```

```

long MPU::getAccelerationZ() {

```

```

    long value;

    Wire.beginTransmission(busAddress);

    Wire.write(MPU_REG_ACCEL_ZOUT_H);

    Wire.endTransmission();

    Wire.requestFrom(busAddress, 2);

    value = Wire.read() << 8 | Wire.read();

    return value;

}

```

MPU.h Code :

Create a file in the Arduino named MPU.h and copy and paste the following lines

```

//

// Mpu Class

```

```
//
```

```
#ifndef MPU_H
```

```
#define MPU_H
```

```
#include "Arduino.h"
```

```
#include <Wire.h>
```

```
#define MPU_REG_CONFIG      0x1A
```

```
#define MPU_REG_GYRO_CONFIG  0x1B
```

```
#define MPU_REG_ACCEL_CONFIG 0x1C
```

```
#define MPU_REG_ACCEL_XOUT_H 0x3B
```

```
#define MPU_REG_ACCEL_XOUT_L 0x3C
```

```
#define MPU_REG_ACCEL_YOUT_H 0x3D
```

```
#define MPU_REG_ACCEL_YOUT_L 0x3E
```

```
#define MPU_REG_ACCEL_ZOUT_H 0x3F
```

```
#define MPU_REG_ACCEL_ZOUT_L 0x40
```

```
#define MPU_REG_GYRO_XOUT_H  0x43
```

```
#define MPU_REG_GYRO_XOUT_L  0x44
```

```
#define MPU_REG_GYRO_YOUT_H  0x45
```

```
#define MPU_REG_GYRO_YOUT_L  0x46
```

```
#define MPU_REG_GYRO_ZOUT_H  0x47
```

```
#define MPU_REG_GYRO_ZOUT_L  0x48
```

```
#define MPU_REG_PWR_MGMT_1    0x6B
```

```
#define RAD2DEGREE            57.296;
```

```
class MPU
```

```
{
```

```
public:
```

```
    MPU(int address);
```

```
    void initialise();
```

```
    void getGyroCalibrationValues();
```

```
    float getAccelAngle();
```

```
    void setTiltAngle(float angle);
```

```
    float getTiltAngle();
```

```

private:

    int accelCalibrationValue = -7586; // Enter the accelerometer calibration value // YABR yazılımından aldığım rakam//orijinal değer -187
// ilk robot -7981 ile ÇALIŞTI !!!

    int busAddress;

    long gyroYawCalibrationValue;

    long gyroPitchCalibrationValue;

    float tiltAngle;

private:

    void reset();

    void wakeup();

    void setGyroFullScaleRange();

    void setAccelFullScaleRange();

    void setLowPassFilter();

    long getRotationX();

    long getRotationY();

    long getRotationZ();

    long getAccelerationX();

    long getAccelerationY();

    long getAccelerationZ();

};

#endif

```

PID.ccp Code :

Create a file in the Arduino named PID.ccp and copy and paste the following lines

```

//

// PID Class

//

#include "Arduino.h"

#include "PID.h"

PID::PID() {

```

```

iMemory = 0;

lastError = 0;

outputMin = 0;

outputMax = 100;

deadband = 0;
}

void PID::reset() {

    output = 0;

    iMemory = 0;

    selfBalanceSetpoint = 0;
}

void PID::setTuningParameters(float p, float i, float d) {

    pTerm = p;

    iTerm = i;

    dTerm = d;
}

void PID::setDeadband(float db) {

    deadband = db;
}

void PID::setOutputLimited(float opMin, float opMax) {

    outputMin = opMin;

    outputMax = opMax;

    iMemory = constrain(output, outputMin, outputMax);
}

float PID::calculate() {

    float error;

    error = processVariable - setpoint - selfBalanceSetpoint;

    // Brake function

    if (output > 10 || output < -10) {

        error += output * 0.015;
    }
}

```

```

iMemory += iTerm * error;

iMemory = constrain(iMemory, outputMin, outputMax);

output = pTerm * error + iMemory + dTerm * (error - lastError);
output = constrain(output, outputMin, outputMax);

lastError = error;

if (output < deadband && output > -deadband) {
    output = 0;
}
}

```

PID.h Code :

Create a file in the Arduino named PID.h and copy and paste the following lines

```

//
// PID Class
//

#ifndef PID_H
#define PID_H

#include "Arduino.h"

class PID
{
public:
    PID();

    void reset();

    void setTuningParameters(float p, float i, float d);

    void setDeadband(float db);

    void setOutputLimited(float opMin, float opMax);

    float calculate();

private:

```

```

float pTerm;

float iTerm;

float dTerm;

float iMemory;

float lastError;

float outputMin;

float outputMax;

float deadband;

public:

float setpoint;

float processVariable;

float output;

float selfBalanceSetpoint;

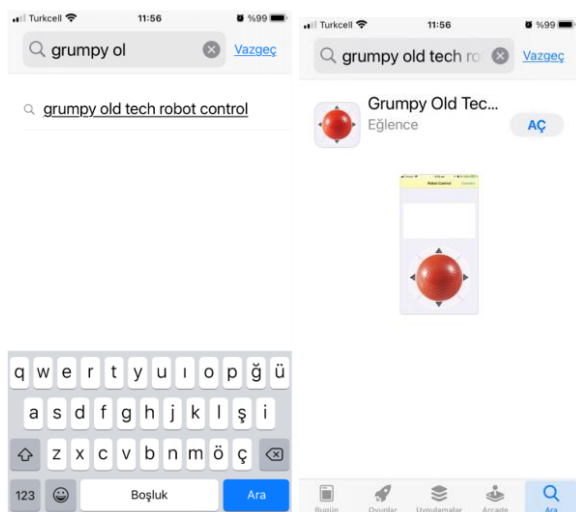
};

#endif

```

8-Downloading i-phone app for controlling robot (from Apple Store)

Download “grumpy old tech robot control” app to your i-phone from Apple Store



9-Starting the Robot

Remove HM-10 module and Upload the Arduino Code to the Robot. It is must to keep HM-10 module be removed while uploading Arduino the code.

After uploading the code the robot will be ready to go. Drive it with your i-phone...ENJOY !!!

