Shaky

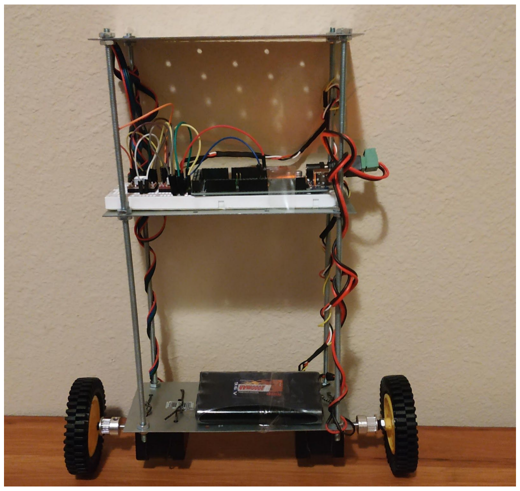
The Self-Balancing Robot

Designed by Group 2:
Jerahmy Bindon, Enbain Kuang, David Landry, and Surafel Mamo
For CSS 427A, Winter 2019
University of Washington, Bothell

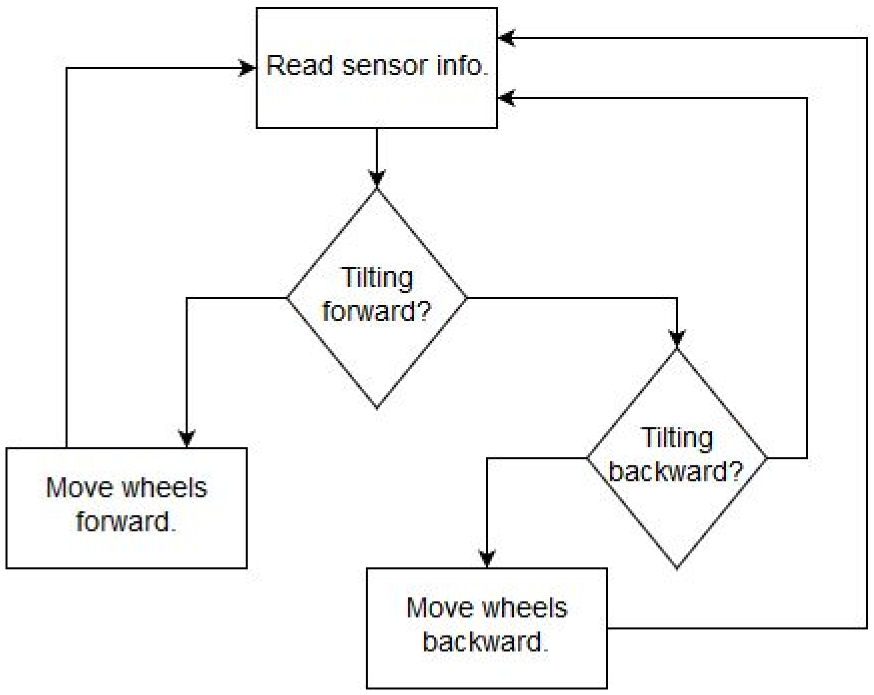
Table of Contents

Introduction

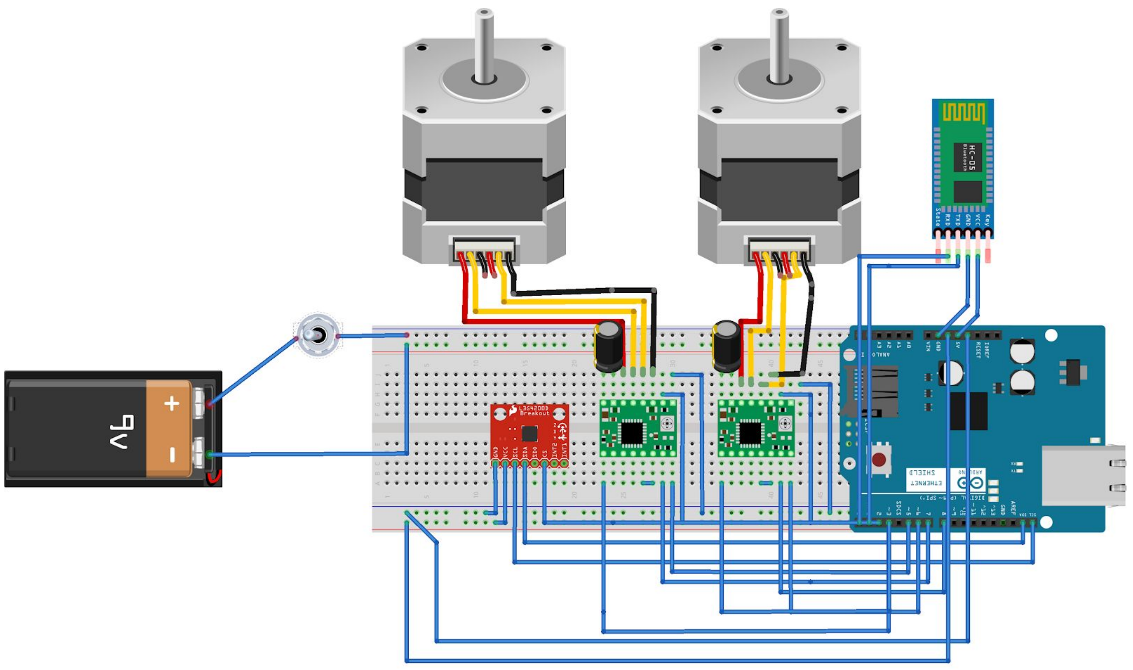
Meet Shaky, the self-balancing robot. Shaky consists of three platforms stacked vertically, mounted on two wheels. The bottom platform houses the battery. The middle platform houses the control unit and accelerometer. The top platform is open for placing objects upon, such as a refreshing beverage or a house of cards.



Control Flow



Parts Schematics



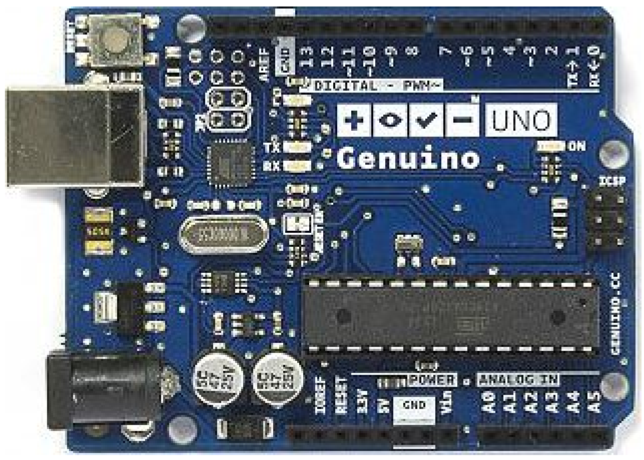
Part 1: Parts List

Structure

Shaky's structure consists of 3 metal platforms mounted on 4 screws. The platforms are secured by 4 pairs of bolts for each platform. The wheels are mounted to two motors secured to the bottom of the bottom platform, additional fasteners and zipties are used as cable management.

Control Unit

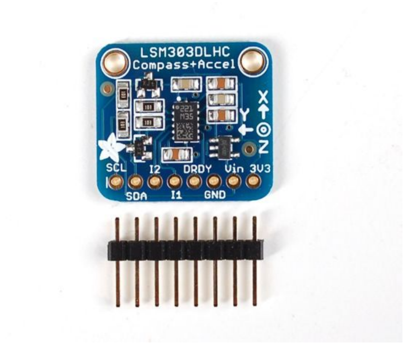
Shaky's control unit is a Genuino Uno, powered by an ATMega328P, an 8-bit AVR microcontroller (https://store.arduino.cc/usa/arduino-uno-rev3).



Part Name	Manufacture	Model	Part Number	Link
Genuino Uno	Atmel	Uno	A000066	https://store.arduino.cc/usa/arduino-uno-rev3

Accelerometer

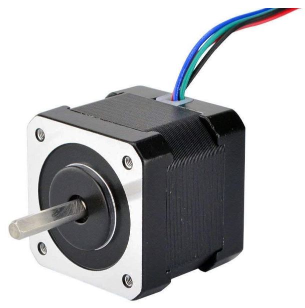
Tilt is measured by an AdaFruit LM303 accelerometer & magnetometer sensor.



Part Name	Manufacture	Model	Part Number	Link
Triple-axis Accelerometer	ST	LSM-303	1120	https://www.adafruit.com/product/1120

Motors

The two stepper motors that are used to move the robots are 2x Nema 17 bipolar stepper motors. This motors have 1 Nm for torque, powerful enough to move and accelerate the base of the robot along with the additional weight added on. Each motors requires an external power of 8-15V at 1000mA to operate. Using the four wires, a stepper driver shield or stepper driver chips can be used to control the precision and direction of the motor.



Part Name	Manufacture	Model	Part Number	Link
Stepper Driver	NIMA	17	2267	https://www.pololu.com/product/2267

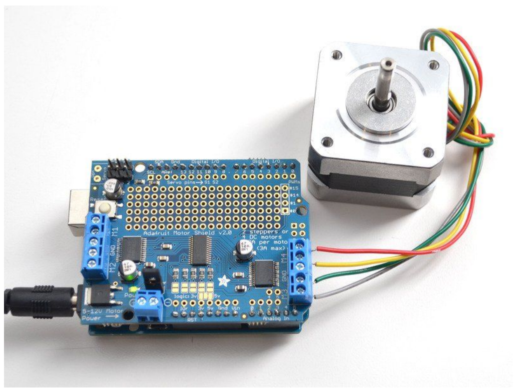
Wheels

Shaky uses a pair of 80 mm rubber wheels.

Part Name	Manufacture	Model	Part Number	Link
Rubber Wheels	DRF-robotic	80mm	RB-Dfr-543	https://www.robotshop.com/en/dfrobot-80mm-rubber-wheel-pair.html

Motor Shield

The two motors and the accelerometer are attached to an AdaFruit Motor Shield, V2. Information on the Motor Shield was referenced at AdaFruit's website (https://www.adafruit.com/product/1438).



Part Name	Manufacture	Model	Part Number	Link
Adafruit Motor/Stepper/ Servo Shield	Adafruit	Motor Shield V2	1438	https://www.adafruit.com/product/1438

Motor Controllers

Previous iterations of Shaky used AdaFruit Motor Shield as the primary driver for controlling each of the two independent motors. Upon implementation, the AdaFruit Motor Shield can only control one stepper motor at the time,making the self balancing maneuver very difficult to achieve. To get around this problem, we upgraded to stepper driver control units that are designed to interface with each motor.

The Pololu A4988 Stepper Motor Driver are small break out boards that contains a chips to controls stepper motors with variable speed and precision. They are designed to interface with external microcontrollers such as arduino to command the position and direction of each of the stepper motors located in the base of the robot.

Part Name	Manufacture	Model	Part Number	Link
Stepper Motor Driver	Pololu	A4988	1182	https://www.pololu.com/product/1182

Battery

A 9.6V battery that provides 2000mA is attached to a 2.1mm center-positive plug adapter is used to power Shaky. This battery helps to run the arduino as well as the various sensors and motors that shaky needs to operate.

Part Name	Manufacture	Model	Part Number	Link
Battery Pack	TYCO	R/C - 9.6V	1182	https://www.amazon.com/Tyco-9-6v-Turbo-Premium-Battery/dp/B0013L9IP8

Part 2: Software

Code

The Arduino IDE was used for all coding for Shaky. The complete source code can be found in the appendix at the end of this document.

Main Algorithm
The main algorithm used for Shaky is a sense-compute-control model. The sensor detects forward or backward acceleration. From there, compensation is calculated. This is multiplied by a multiplier, which controls the speed at which the tires rotate to compensate for tipping. Step motors operate by stepping via magnets. The algorithm adjusts the step magnitude based on the tilt recorded by the accelerometer.

Libraries

To properly interface with the Motor Shield, the AdaFruit Motor Shield v2 library was downloaded and implemented. The library is searchable from within the Arduino IDE. The library includes functions that instantiate a motor shield object, and instantiate & control stepper motors.

To interface with the LSM303 sensor, two libraries were downloaded: Adafruit_Sensor and Adafruit_LSM303DLHC, both searchable from the Arduino IDE.

Part 3: Conclusion

Project Status and Challenges

While building different iteration of Shaky, we were faced with many obstacles we did not foresee. Some of the core challenges include hardware issues and software integration. The first iteration of the robot were inoperational due to the speed of the motor. We original selected a DC motor drive that was suited to drive remote controlled vehicles. Upon implementation, we discovered speed of the motor were not adequate enough to maintain the balancing maneuver.

Since our original premise was incorrect, we decided to upgrade the motors with faster RPM to helps us achieve our goal. Following the instruction set by the adafruit motor shield, we connected the newly update motors to our drive and discovered their output was not sufficient enough to hold shalky weight. The added batteries, sensors and metal frame has added an extra weight to our original design, making shaky much heavier than before. In the last attempt and careful research, we found the nema 17 steppers motors that had enough torque to maintain and accelerate the robots bodie of mass.

The second major challenge we faced was the interface between the microcontroller and the stepper motor drivers. The stepper driver require an external power supply in addition to a logic power to maintain the motor chip. Having power issues made the communication between the microcontroller and stepper driver very buggy and therefore had issues executing the proper maneuver to help the robot stay balanced.

Future Improvements

- To improve control precision in the future, a fuzzy logic table could be used to define tire speed and direction.
- Current PID system can be refined better to help with the balance
- Future iterations will allow bluetooth control. Subsumption architecture would be used, with collision avoidance at the second level and bluetooth control at the third level of subsumption. Supporting these capabilities would involve the addition of a bluetooth unit and a sonar sensor.

Appendix: Source Code

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_LSM303_U.h>
#include <Adafruit_MotorShield.h>
#include <utility/Adafruit_MS_PWMServoDriver.h>

double arx, ary, arz, grx, gry, grz, gsx, gsy, gsz, rx, ry, rz;
double gyroScale = 131;
int16_t ax, ay, az, gx, gy, gz
float Kp, Ki , Kd , Kps ,Kis ,Kds;
float integralSum, PID_errorSum;
float errorAnterior, errorAnt;

// Create a Motor Shield object
//Adafruit_MotorShield AFMS = Adafruit_MotorShield();

// Create motor objects
//Adafruit_DCMotor* leftMotor = AFMS.getMotor(1);
//Adafruit_DCMotor* rightMotor = AFMS.getMotor(3);

/* Assign a unique ID to this sensor at the same time */
//Adafruit_LSM303_Mag_Unified mag = Adafruit_LSM303_Mag_Unified(12345);
/* Assign a unique ID to this sensor at the same time */
//Adafruit_LSM303_Accel_Unified accel = Adafruit_LSM303_Accel_Unified(54321);

void setup(void)
{
  Serial.begin(115200);
  Serial.println("Magnetometer Test"); Serial.println("");

  // Setup the Motor Shield
  AFMS.begin();
  leftMotor->setSpeed(0);
  rightMotor->setSpeed(0);

  /* Initialise the sensor */
  if(!mag.begin())
  {
    /* There was a problem detecting the LSM303 ... check your connections */
    Serial.println("Ooops, no LSM303 detected ... Check your wiring!");
    while(1);
  }

  if(!accel.begin())
  {
    /* There was a problem detecting the ADXL345 ... check your connections */
    Serial.println("Ooops, no LSM303 detected ... Check your wiring!");
```

```
   while(1);
  }

   pinMode(4,OUTPUT);  // STEP MOTOR 1 PORTD,4
   pinMode(5,OUTPUT);  // DIR MOTOR 1  PORTD,5
   pinMode(7,OUTPUT);  // STEP MOTOR 2 PORTD,7
   pinMode(6,OUTPUT);  // DIR MOTOR 2  PORTD,6
   pinMode(8,OUTPUT);  // ENABLE PORTB,0
   digitalWrite(8,LOW);

   TCCR1A = 0;                                    // Timer1 CTC mode 4
   TCCR1B = (1<<WGM12)|(1<<CS12);                        // Prescaler=256
   OCR1A = 255;
   TCNT1 = 0;

   TCCR2A = (1<<WGM21);                                  // Timer2 CTC mode 4
   TCCR2B = (1<<CS22)|(1<<CS21);                        // Prescaler=256
   OCR2A = 255;
   TCNT2 = 0;

   dir_M1 = 0;
   dir_M2 = 0;

}

void loop(void) {
  /* Get a new sensor event */
  sensors_event_t event;
  mag.getEvent(&event);
  accel.getEvent(&event);
  Serial.print("X: "); Serial.print(event.acceleration.x); Serial.print("  "); Serial.println("m/s^2 ");
  float compensation = event.acceleration.x * -1;

    if ((anguleActual<55)&&(anguleActual>-55)){
     if (anguleActual<(anguleObjetivo+H) && anguleActual>(anguleObjetivo-H) && (velocity<V &&
velocity>-V))  {
       CLR(PORTB,0);
       vel1 = 0 + giro ;
       vel2 = 0 - giro ;
       integralSum = 0;
       setMotorSpeedM1(vel1);
       setMotorSpeedM2(vel2);
     }  else {
        CLR(PORTB,0);
        velocity = resultadoPID;
        vel1 = velocidad + giro ;
        vel2 = velocidad - giro ;
        vel1 = constrain(vel1,-VEL_MAX,VEL_MAX);
        vel2 = constrain(vel2,-VEL_MAX,VEL_MAX);
        setMotorSpeedM1(vel1);
        setMotorSpeedM2(vel2);
       }
   }else {
        SET(PORTB,0);
        setMotorSpeedM1(0);
        setMotorSpeedM2(0);
        integralSum = 0;
        PID_errorSum = 0;
      }

}

int calcVel (float x){
    float mini = 210;
    float vel = 20623.333333333333;
    vel = vel / 500;
    vel = (vel * x) + mini;
    vel = 1 / vel;
    vel = vel / 0.000016;
    return ((int)vel-1);
}

void setMotorSpeedM1(int tspeed){
    int16_t speed1;
    if ((speed_M1 - tspeed)>MAX_ACCEL)
        speed_M1 -= MAX_ACCEL;
    else if ((speed_M1 - tspeed)<-MAX_ACCEL)
        speed_M1 += MAX_ACCEL;
    else
        speed_M1 = tspeed;
    speed1 = speed_M1;
    if (speed1==0) {
        timer_period1 = ZERO_SPEED;
        dir_M1 = 0;
    } else if (speed1>0) {
        timer_period1 = calcVel(speed1);
        dir_M1 = 1;
        SET(PORTD,5);
    } else {
        timer_period1 = calcVel(-speed1);
        dir_M1 = -1;
        CLR(PORTD,5);  // Dir Motor 1
    }
    if (timer_period1>255) {timer_period1 = ZERO_SPEED;}
    OCR1A = timer_period1;
    if (TCNT1 > OCR1A)
        TCNT1 = 0;
}

void setMotorSpeedM2(int tspeed) {
    int16_t speed2;
    if ((speed_M2 - tspeed)>MAX_ACCEL)
        speed_M2 -= MAX_ACCEL;
    else if ((speed_M2 - tspeed)<-MAX_ACCEL)
        speed_M2 += MAX_ACCEL;
    else
```

```
        speed_M2 = tspeed;
    speed2 = speed_M2;

    if (speed2==0){
        timer_period2 = ZERO_SPEED;
        dir_M2 = 0;
    } else if (speed2>0) {
        timer_period2 = calcVel(speed2);
        dir_M2 = 1;
        CLR(PORTD,6);
    }
    else {
        timer_period2 = calcVel(-speed2);
        dir_M2 = -1;
        SET(PORTD,6);
    }
    if (timer_period2 > 255)
        timer_period2 = ZERO_SPEED;

    OCR2A = timer_period2;

    if (TCNT2 > OCR2A)
        TCNT2 = 0;
}

float PID (float angleActual, float angleDescent, float Kp, float Kd, float Ki, float time) {
    float errorActual;
    float salida;
    float output;

    errorActual = angleDescent - angleActual;
    float proporcional = errorActual * Kp * 0.1;
    integralSum = integralSum + errorActual ;
    float integral = Ki * integralSum * time * 0.001;
    integral = constrain (integral, -VEL_MAX, VEL_MAX);
    float derivitaive = Kd * gsx / time ;
    errorAnterior = errorActual;
    output = constrain((proporcional - derivitaive  + integral), -VEL_MAX, VEL_MAX);
    return (output);
}
```