Jack the Jumping Robot
Pinion-based springpowered jumping robot

JAKOB JANSSON
IDA LARSSON
Jack the Jumping Robot

Pinion-based spring powered jumping robot

IDA LARSSON
JAKOB JANSSON

Bachelor’s Thesis at ITM
Supervisor: Nihad Subasic
Examiner: Nihad Subasic

TRITA-ITM-EX 2019:39
Abstract

This bachelor thesis will demonstrate and explain the building of a jumping robot. This work took place during the Spring semester of 2019 and was finished in May. The scope was to build a foot sized robot that can regulate jumping force and angle, while maintaining the ability to land safely and with its right side up. This thesis required that the robot made use of sensors and microcontrollers. The electronics used for this robot was an Arduino UNO, an IMU and several servos. The code behind the jump was based on the switch case statement model.

Jack was built with three subsystems in mind: Energy storage to use for jumping, Decoupling for releasing a variable amount stored energy quickly and Suspension for reducing impact on landing and to prevent tipping.

The work resulted in a robot who could jump 6.5% of its body length, regulate jump power and land on its feet. In conclusion, even though some of the subsystems were successful, our model of an jumping robot is insufficient due to its low jump height.

Keywords: Mechatronics, Jumping, Robot, Arduino.
Referat

Den hoppande roboten Jack


Jack är byggd med tre delsystem i åtanke: Energilagring för hoppet, Ackoppling för att avlösa en varierande mängd lagrad energi snabbt och Dämpning för att reducera stötarna vid landing och för att förhindra att roboten välter.

Arbetet resulterade i en robot som kan hoppa 6.5% av sin längd, regula hoppkraften och landa utan att välta. Vi kom fram till, även om vissa delsystem var lyckade, att vår modell av en hoppande robot inte är tillräcklig på grund av sin låga hopphöjd.

Nyckelord: Mekatronik, Hopp, Robot, Arduino.
Acknowledgements

First, we would like to thank Nihad Subasic for organizing seminars and performing lectures to aid our process. We would also like to thank Staffan Qvarnström for overseeing our workshops and providing electrical components. Thanks to Tomas Östberg for providing us with mechanical components such as springs, nuts and bolts. We would like to thank our fellow classmates during this course for listening and giving input on our seminars and on our report. And lastly, a huge thanks to all the assistants, in particular Seshagopalan Thorapalli Muralidharan and Sresht Iyer, for helping our projects with very helpful input and knowledge.
Contents

1 Introduction ................................. 1
  1.1 Background .................................. 1
  1.2 Purpose .................................. 1
  1.3 Scope .................................. 2
  1.4 Method .................................. 2

2 Theory ........................................ 3
  2.1 Arduino .................................. 3
  2.2 Inertial Measurement Unit ............. 3
  2.3 Servo motors ............................... 3
  2.4 Jump height ................................ 4

3 Demonstrator ................................ 5
  3.1 Hardware .................................. 5
    3.1.1 Main body ................................ 5
    3.1.2 Energy storage subsystem .......... 5
    3.1.3 Decoupling subsystem ............... 6
    3.1.4 Suspension subsystem ............... 7
  3.2 Electronics ................................ 8
    3.2.1 Microcontroller ....................... 8
    3.2.2 Prototype perfboard ................ 8
    3.2.3 Buttons ................................ 8
    3.2.4 Servo motors ......................... 8
    3.2.5 IMU .................................. 8
  3.3 Software .................................. 9

4 Results ...................................... 11
  4.1 Jump height ................................ 11
  4.2 Acceleration and angular velocity ..... 12

5 Discussion and conclusions ................. 13
  5.1 Research questions ....................... 13
    5.1.1 Acceleration and angle velocity .. 13
    5.1.2 Suitable design ....................... 13
## List of Figures

3.1 A rendered picture of the whole robot, made in Solid Edge and Keyshot. 

3.2 A view on the robot from below, made in Solid Edge and Keyshot.

3.3 The disconnection mechanism for the pinion servo. To the left is an image of an attached pinion, to the right is an image of an detached pinion. Made in Solid Edge and Keyshot.

3.4 The suspension system that reduces the impact on landing, made in Solid Edge and Keyshot.

3.5 All servos on the robot, left shoulder servo is obstructed by the Arduino UNO. Made in Solid Edge and Keyshot.

3.6 The x-, y- and z-axis of the IMU and the placement of the component on the robot. Made in Solid Edge and Keyshot.

3.7 The flowchart that illustrates the Arduino code. Made in Lucidchart.

4.1 The robot at its maximum jump height with a line drawn to show the jump height.

4.2 The acceleration of the robot during a jump, plotted using Matlab.

4.3 The angular velocity of the robot during a jump, plotted using Matlab.
Nomenclature

Glossary

I2C  short for Inter-Integrated Circuit. It is used to connect lower-speed processors to the microcontroller. 8, 13

Acronyms

$E$  Potential energy. 4
$g$  Gravitational constant. 4
$k$  Spring constant. 4
$m$  Mass. 4

$\text{CO}_2$  Carbon Dioxide. 1

IDE  Integrated Development Environment. 3
IMU  Inertial Measurement Unit. 2, 3, 5, 8, 9, 13
PCB  Printed Circuit Boards. 8
PWM  Pulse Width Modulation. 3, 8
Chapter 1

Introduction

This chapter aims to explain the background and purpose of the project, within what scope it will be carried out in and what method that will be used.

1.1 Background

Having the ability to jump gives robots more ability to traverse tougher terrains and move past more obstacles than just being able to walk or drive. Creating robots that can jump is a big challenge, as jumping requires release of a lot of energy in a short period of time and may also need the ability to control and stabilize the body in the air.

The research and development of hopping mechanism for robots manifested with the increasing interest in outer planetary investigations and celestial exploration[1]. The ability to jump would give rovers more flexible pathing through these hazardous and uneven environments. Though such system would seem useful, it brings multiple complications in order to achieve the preferred robustness of celestial exploration and thus requires more research and development. Another application of hopping robots are speculated to be in disaster rescue robots, who could navigate in complex environments and search for survivors[2].

Boston Dynamics have several robots that can jump and land safely[3][4]. These robots are rather large and heavy. Another notable robot from Boston Dynamics with jumping capabilities is the "Sand flea" that stops and take aim then launches into the air using a piston powered by Carbon Dioxide (CO2)[5]. It flips in the air and does not always land on its wheels, but it has a recovery mechanism to get back on the right side to be able to move around again.

1.2 Purpose

The purpose of the project was to create a robot that can jump up and forward on a flat surface within a controlled environment with no external disturbances. The goals are to be able to control the jump height, starting jump angle and land safely.
CHAPTER 1. INTRODUCTION

This report will discuss and answer the following research questions:

1. How does the acceleration and angle velocity behave during the jumping sequence?

2. Will the design of the robot be sufficient for the intended usage?

1.3 Scope

The robot was planned to be small with a main body no longer than 25 cm long. The main goal of the robot was to be able to regulate jumping force, adjust starting angle and reduce the impact of the landing. Due to time constraints and budget limitations, the whole jump sequence (take off and landing) was hardcoded and no control theory was be applied. Three criteria was needed to be achieved with this thesis, the construction needed components based on: Mechanics, Electronics and Programming. This project required the use of sensors and microcontrollers.

1.4 Method

The robot used a store-and-release method with a spring as a way to jump. Energy is loaded into extension springs as the robot prepares to jump which will then be released to create vertical height and horizontal movement. The front legs of the robot were controlled by servo motors. The front legs were used to control the starting angle of the robot depending on how far and high it was requested to jump. While in the air, the robot moved the legs forward to prepare itself for landing. Jack used an Inertial Measurement Unit (IMU) to be able to document its acceleration and rotation during the jump sequence.

The research questions, see chapter 1.2, will be answered with these methods:

- The acceleration and angular velocity will be measured by an IMU.

- The sufficiency will be answered by comparing jump height to the body size, the reliability of the spring mechanism, the ability to reduce impact and successfully land on its feet.
Chapter 2

Theory

This chapter aims to explain the necessary theory for the reader to understand the demonstrator for the thesis.

2.1 Arduino

Arduino is an open source electronic platform that can be used for a large variety of electronic projects, simple to advanced. The Arduino Software (IDE) is used to program the boards, which is also open-source[6]. In this project an Arduino UNO was used, it comes with six Pulse Width Modulation (PWM) outputs. These pins can vary from 0-5V analog output and can for example regulate motors or LED with 0V to 5V power on their own. There are also 12 digital input/output pins that can read and write HIGH or LOW digital signals. The digital pins can regulate servos with 0 to MAX values, but that would require that the servos have an external power supply[7].

2.2 Inertial Measurement Unit

An inertial measurement unit, IMU, uses a combination of accelerometers and gyroscopes to calculate and measure position, velocity and orientation. Accelerometers can sense tilting motions as well as static and dynamic acceleration (linear acceleration). Gyroscopes on the other hand can sense angular velocity, the rotational rate.

2.3 Servo motors

A servo motor can be divided into three basic components: a motor, a potentiometer and a control board. A potentiometer is a variable resistance that allows the servo to monitor if it is at the correct angular position and compare it to the given value by the control signal. It also has a high power for its small format. This makes the
servo motor a good choice when ease-of-use and high power combined with small weight are requested\[8\].

Discontinuous servo motors have a certain revolution cut-off, such as 180°. This means that those kind of servos have a limited rotation range. Continuous servo motors do not have a revolution cut-off and can spin endlessly in both directions.

2.4 Jump height

A spring is an elastic object that can store elastic potential energy $E$\[9\]. The energy can be calculated with

$$ E = \frac{1}{2} k(x - x_0)^2 $$

(2.1)

where $x$ is the current length of the spring, $x_0$ is the springs equilibrium length and $k$ is the spring constant. Due to gravity, a body of mass has a potential energy $E$ depending on height of position defined by

$$ E = mgh $$

(2.2)

where $m$ is the mass of the body, $g$ is the gravity constant and $h$ is the relative height.

Combining 2.1 and 2.2 and solving for $h$ is the potential jump height using the elastic energy of a spring if no losses between the energy conversion happens

$$ h = \frac{1}{2mg} k(x - x_0)^2 $$

(2.3)
Chapter 3

Demonstrator

A demonstrator was created as a proof-of-concept and for getting test results. This chapter will show the construction and how it works.

3.1 Hardware

The robot can be divided into a main body and three subsystems; Energy storage, Decoupling and Suspension. These subsystems are explained below, see Figure 3.1 for an image of the whole robot. Two small servos with attached legs was used in the "shoulders" of the robot to regulate the robots angle to the ground, meanwhile making the robot be able to lay down and stand up.

3.1.1 Main body

The robot was 250 mm long when the front legs are close to the body and the springs were in their relaxed state. The whole demonstrator weighs 333 g. Most of the demonstrator was made out of acrylic due to its low density (compared to metals), hardness, availability and low friction against other parts of acrylic. The low friction quality was very important since the jumping legs would be sliding against the main body and the hooks keeping the legs flush against the body. A lot of material was cut-out in many parts, such as the main body and the legs, to reduce weight to allow for a higher jump. The main body was also the platform where the Arduino UNO, prototype perfboard and IMU was mounted on.

3.1.2 Energy storage subsystem

The robot jumped by using extension springs to store energy and then release it quickly towards the surface underneath the robot. A gear rack and pinion attached to a servo motor can together pull on the springs, see Figure 3.2. Linear motion can be achieved with the pinion connected to a gear rack. It was then possible to convert the torque of the motor to a force pulling on the extension springs.
3.1.3 Decoupling subsystem

The robot would not jump if the pinion stayed connected to the pinion due to the internal gears of the servo. It needed to decouple fast for a large impulse force. This was done by placing the pinion on a movable arm, together with a decoupling servo, see Figure 3.5. The decoupling servo pushed against a small pin to rotate the entire arm and thus the servo with the pinion from the gear, see Figure 3.3. This arm had a connected extension spring to keep the pinion pressed against the gear rack at all times, except when the decoupling servo was activated to decouple.
3.1. HARDWARE

Figure 3.3. The disconnection mechanism for the pinion servo. To the left is an image of an attached pinion, to the right is an image of an detached pinion. Made in Solid Edge and Keyshot.

3.1.4 Suspension subsystem

In order to reduce the impact of landing, a suspension system was installed. It worked similar to a car suspension by having the two front shoulder servos be mounted on pivoting arms, two compressing springs were placed between the servos and the main body and thus acted like a cushioning, see Figure 3.4. This design was used because it had a secondary effect. When the springs compressed, both arms would tilt upwards, increasing the distance between the tips of the legs. This provided a wider landing surface to reduce the possibility for the robot to tumble sideways and fall over when landing.

Figure 3.4. The suspension system that reduces the impact on landing, made in Solid Edge and Keyshot.
3.2 Electronics

This section will explain the electrical components used in this project. For a schematic picture of the wiring, see the attached document in Appendix A.

3.2.1 Microcontroller

Arduino UNO was used as the microcontroller for this demonstrator. It was used due to being lightweight coupled with ease-of-use. It provided useful properties such as I2C protocol for the MPU-6050 (IMU), PWM and digital pins for all the servos and buttons.

3.2.2 Prototype perfboard

A standardized board with small holes covered with thin copper plates, this was used for soldering our circuit without the use of an Printed Circuit Boards (PCB).

3.2.3 Buttons

Regular buttons where used for triggering switch case statements manually on the robot. They where connected with a pull-up setup to the Arduino UNO. Pull-up is a setting in the Arduino Uno code which uses its internal resistors to prevent the inputs to trigger from noise in the current. This removes the need to solder resistors to the circuit.

3.2.4 Servo motors

There were four servo motors used in total for the robot. Two for its front legs (shoulder servos), one for for pulling the extension springs so it can jump (pinion servo) and the last one for moving the pinion servo from the rack (decoupling servo), see Figure 3.5.

The two shoulder servos were small, discontinuous servos of the same model. The decoupling servo were larger than the shoulder servos, because it needed to work with higher torque. The pinion servo, the one that needs to pull on the four extension springs, needed to be stronger and had a much higher torque and size.

3.2.5 IMU

The IMU, MPU-6050, was used to get the acceleration and rotation during a jump. It has a three-axis accelerometer sensor, a three-axis gyroscope sensor and a processor that can process the rawdata from the sensors. It communicates via I2C with the microcontroller.

The IMU is mounted on the robot just underneath the Arduino UNO. This means the data of each axis given by the IMU were relative its position on the robot,
3.3. SOFTWARE

not relative the surface it was jumping from. The axis orientation and placement can be seen in Figure 3.6.

**Figure 3.5.** All servos on the robot, left shoulder servo is obstructed by the Arduino UNO. Made in Solid Edge and Keyshot.

**Figure 3.6.** The x-, y- and z-axis of the IMU and the placement of the component on the robot. Made in Solid Edge and Keyshot.

3.3 Software

The robot was controlled by switch case statements. States are several instructions combined in one section and will only run if the state is switched to. This provided useful interactions when programming and demonstrating different jumps and components. The flowchart is represented in Figure 3.7. The states were controlled by buttons mounted on the robots prototype perfboard.

The code used for the robot can be found in Appendix B. A separate code was used for getting jumping data from the IMU, which can be found in Appendix C. Appendix C used the same setup loop and Case 1 (the whole jump sequence) from the normal program, with some added lines that read data from the IMU.
Figure 3.7. The flowchart that illustrates the Arduino code. Made in Lucidchart.
Chapter 4

Results

This chapter aims to present results from the demonstrator. The robot was placed on a flat, hard surface for all tests.

4.1 Jump height

The robot was filmed while it was jumping to see what maximum jump height that was achieved. The "foot" closest to the camera was aligned with the ruler so that the perspective would not be warped to get more accurate results. The robot reached a max jump height at about 16 mm reliably. See Figure 4.1 for a picture during a jump with a ruler beside.

![Figure 4.1. The robot at its maximum jump height with a line drawn to show the jump height.](image-url)
CHAPTER 4. RESULTS

4.2 Acceleration and angular velocity

During testing, the robot never tumbled and did not fall over sideways after landing. It jumped almost completely straight forward.

A jump was repeated seven times, with the pinion rotating the same speed and time for every jump attempt to get data from the IMU. Data points for each axis were taken every 100 milliseconds. This was the shortest time between data points that was achieved reliably in this project. All data can be found in Appendix D.

The median for every data point of the same corresponding time in the jump sequence is illustrated for each axis in Figure 4.2 for acceleration and Figure 4.3 for angle velocity. The x-, y- and z-axis is in the same orientation found in Figure 3.6.

![Figure 4.2. The acceleration of the robot during a jump, plotted using Matlab.](image1)

![Figure 4.3. The angular velocity of the robot during a jump, plotted using Matlab.](image2)
Chapter 5

Discussion and conclusions

This chapter will discuss the research questions, future work and final conclusions.

5.1 Research questions

This section will discuss the established research questions mentioned in the Introduction.

5.1.1 Acceleration and angle velocity

This section aims to answer the first research question:

1. How does the acceleration, angle velocity and rotation behave during the jumping sequence?

The IMU can only send data every 100 ms with the method used in this project. This could also be due to communicating with I2C between the Arduino and the IMU. A jump only lasted for a few 100 ms, resulting in only getting very few data points during the jump, see Figure 4.2 and 4.3.

As can be seen in the 4.3 and 4.2, the robot had some angle velocity around the x-axis and acceleration along the y-axis. This was because the servos were not calibrated correctly which made it difficult to have the front legs parallel. This resulted in the robot not jumping entirely straight but Jack was still stable during the jump as it did not fall to the side during landing.

5.1.2 Suitable design

This section aims to answer the second research question:

2. Will the design of the robot be sufficient for the intended usage?

This will be answered by analyzing the subsystems of the robot, starting with the Energy storage subsystem. The robot managed to fit four extension springs into
its main body, the springs where inserted into cut-out tracks inside both the main body and the back leg to keep them in place. These springs could be switched to different ones with a bigger or smaller spring constant, but ultimately the current ones were chosen based on the sturdiness of the body. This resulted in a maximum jump height of 16 mm. Although the jump was small, this subsystem was functional in its design because of the placement and support of different types of springs.

Next is the Decoupling subsystem. It succeeded in the ability to manually release the stored energy, thus allowing the robot able to regulate the power behind the jump. Jack’s maximum jump was not very large, so this function did not serve any useful purpose. The coupling also had a fail rate of about 15% due to the pinion separating from the gear-rack. The spring that kept the pinion and gear-rack connected did have enough strength to hold these two together when maximum jump power was loaded. But the robot but would sometimes lose the connection when the pinion servo was spinning. This may have been a result of the pinion not being perfectly parallel to its rotation axis, where two teeth would occasionally meet and separate the gear-rack and pinion. Given the high fail rate, this subsystem was not sufficient for a jumping robot.

Lastly, the Suspension subsystem was difficult to analyze because of the low jumping height. Given the robots maximum jump height, this system was successful in reducing the impact of the landing. During testing the robot did not tumble once, which lead to the conclusion that this system worked as intended. When analyzing the videos of the jump sequence, it was noted that the suspension springs compressed about halfway on landing. This lead to the conclusion that this system was sufficient and necessary for a jumping robot.

5.2 Future work

This section will attempt to discuss how the results of this project can be used for future projects.

The scalability of this project is difficult to predict. If the structural dimensions of the robot increase, its sturdiness will become exponentially stronger based on solid mechanics. If the amount of springs and their spring constant increase, a higher jump height might be possible. The question is if this will overpower the negative effect of increasing the material and weight of the robot, according to equation 2.3, the jump height is inversely proportional to the mass. This project would have relatively better results if it had a larger scale. With the use of aluminum or carbon fibre this robot could wield larger springs and have a more detailed construction that empower sturdiness.

Metal bearings and tracks were avoided in order to reduce weight. With the relative size of the robot, metal components were heavy. If the dimensions increase, this relative aspect will be less dominant and a lot of friction and loss of energy could be reduced.

The Decoupling subsystem can drastically be improved on. It did not, unlike the
5.3. CONCLUSIONS

*Energy storage subsystem*, have its active forces on the same plane. This created sideways torque that twisted the whole movable arm and was most likely the cause of the pinion not being perfectly parallel to its rotation axis.

In reality, this robot would need several more subsystems to be useful such as; power supply, rotational positioning in the horizontal plane, cameras, feedback loops and tools for gathering data on its surroundings. These systems would increase the weight drastically.

5.3 Conclusions

The robot achieved a jump of 6.5% of its body length. The *Energy storage subsystem* worked in practice, but ultimately did not have enough power to jump high enough. The *Decoupling subsystem* worked for varying the amount of energy stored in the springs and therefore jump capability, but it didn’t have much of a practical use for this robot due to its small maximum jump height. The *Suspension subsystem* reduced impact of the landing and prevented the robot from falling over.

The robot did not jump completely straight due to the servos not being correctly calibrated. The robot still lands safely and the robot did not tumble in the air.

A lot of work was put into reducing the weight of the robot. Servos were carefully picked to precisely manage its load, reducing its weight to a minimum. The acrylic body had cut-out holes and components made of metal where kept to a minimum.

In summary, the design of Jack in its current stage is not sufficient for a jumping robot. This field is relatively unexplored and is in need of research. This project might provide useful input and ideas for the future development of jumping robots.
Bibliography


[9] Nicholas Apazidis, *Mekanik II: partikelsystem, stel kropp och analytisk mekanik*  
Stockholm: Studentlitteratur AB, 2012
Appendix A

Wiring

The wiring used on the robot. The picture was made in Fritzing.
Appendix B

Arduino Code - Main program

The full source code uploaded to the Arduino.

/*
This is the main code for the thesis work "Jack the jumping robot".

Created by: Ida Larsson and Jakob Jansson
Finished: 4th of May 2019
*/

#include <Servo.h>

Servo decouple; // The small servo that makes the pinion
                 // disconnect from the gear rack
Servo pinion;   // The continuous servo that turns the pinion
Servo leftshoulder; // The robot's left front shoulder
Servo rightshoulder; // The robot's right front shoulder

int leftstart = 130; // The starting angle in degrees
int rightstart = 125 - leftstart; // These are invers of eachother
int leftend = 0;  // The ending angel in degrees
int rightend = 130 - leftend;
int att = 40;     // The pinion is attached to the gear rack
int det = 80;     // The pinion is detached from the gear rack
int pinionspin = 2000; // The amount of milliseconds of activation
int inpin1 = 8;   // Set pin
int inpin2 = 9;   // Set pin
int inpin3 = 10;  // Set pin
int inpin4 = 11;  // Set pin
int button = 0;   // Set button
int stand = 0;    // Set stand
APPENDIX B. ARDUINO CODE - MAIN PROGRAM

```cpp
void setup() {
    // this section runs only once
    decouple.attach(2); // Assign to pin
    pinion.attach(3);  // Assign to pin
    leftshoulder.attach(4); // Assign to pin
    rightshoulder.attach(5); // Assign to pin
    pinMode(inpin1, INPUT_PULLUP); // Pullup to avoid noise activation
    pinMode(inpin2, INPUT_PULLUP); // Pullup to avoid noise activation
    pinMode(inpin3, INPUT_PULLUP); // Pullup to avoid noise activation
    pinMode(inpin4, INPUT_PULLUP); // Pullup to avoid noise activation
    pinion.write(85); // Mid point for continuous servo
    decouple.write(att); // Attach the pinion
    leftshoulder.write(leftend); // Lay down
    rightshoulder.write(rightend); // Lay down
}

void loop() {
    int button1 = digitalRead(inpin1); // Read button 1
    int button2 = digitalRead(inpin2); // Read button 2
    int button3 = digitalRead(inpin3); // Read button 3
    int button4 = digitalRead(inpin4); // Read button 4
    if (button1 == LOW) { // Assign case
        button=1;
    }
    if (button2 == LOW) { // Assign case
        button=2;
    }
    if (button3 == LOW) { // Assign case
        button=3;
    }
    if (button4 == LOW) { // Assign case
        button=4;
    }
    switch (button) {
    case 1: // Jump sequence + landing
        delay(2000); // Time for removing finger from button
        pinion.write(67); // The continuous servo is spinning
        delay(pinionspin); // Spin during this time
        pinion.write(85); // Stop the pinion servo
        break;
```
leftshoulder.write(leftstart);  // Stand up
rightshoulder.write(rightstart); // Stand up
delay(1500);                   // Wait 1.5 seconds

decouple.write(det);          // Release and jump
delay(50);                    // Delay 50 milliseconds
leftshoulder.write(leftend);  // Stretch the legs
rightshoulder.write(rightend); // Stretch the legs
delay(1000);                  // Wait 1 second
decouple.write(att);          // Attach the cog

delay(1000);                  // Delay 1 second
button = 0;                   // Reset case
break;                        // Leave the case

case 2:                        // test detach-mechanism

delay(1000);                  // Time to remove finger from button
decouple.write(det);         // Detach the rack and pinion
delay(1000);                  // Wait 1 second
decouple.write(att);         // Attach the rack and pinion

delay(1000);                  // Delay 1 second
button = 0;                   // Reset case
break;                        // Leave the case

case 3:                        // Toggle stand up/lay down

delay(1000);                  // Time to remove finger from button
if (stand == 1) {             // If it is standing up
leftshoulder.write(leftend); // Lay down
rightshoulder.write(rightend); // Lay down
stand = 0;                    // Remember laying down
}
else {                        // If it is laying down
leftshoulder.write(leftstart); // Stand up
rightshoulder.write(rightstart); // Stand up
stand = 1;                    // Remember standing up
}

delay(1000);                  // Wait 1 second
button = 0;                   // Reset case
break;                        // Leave the case
case 4: // Unused case
    delay(100);
}
delay(50); // Refreshrate of the loop
Appendix C

Arduino Code - IMU data

The full source code uploaded to the Arduino to get accelerometer and gyroscope readings during a jump.

/∗
This is the code for reading acceleration and rotation
for the thesis work "Jack the jumping robot".
∗/

Created by: Ida Larsson and Jakob Jansson
Finished: 4th of May 2019
*/

#include <Wire.h>
#include <Servo.h>

Servo decouple; // The small servo that makes the pinion
                // disconnect from the gear rack
Servo pinion;   // The continuous servo that turns the pinion
Servo lefthoulder; // The robot’s left front shoulder
Servo rightshoulder; // The robot’s right front shoulder

const int MPU6050_addr=0x68;  // The I2C address

//int16_t AccX,AccY,AccZ,Temp,GyroX,GyroY,GyroZ;

float AccX,AccY,AccZ,Temp,GyroX,GyroY,GyroZ;

int leftstart = 130;       // The starting angle in degrees
int rightstart = 125 - leftstart;       // These are invers of eachother
int leftright = 0;         // The ending angle in degrees
int rightend = 130 - leftend;
int att = 40;              // The pinion is attached to the gear rack
int det = 80; // The pinion is detached from the gear rack
int pinionspin = 2000; // The amount of milliseconds off activation
int inpin1 = 8; // Set pin, used for button
int button = 0; // Set button
int debug = 0; // For debugging purposes
int delayT = 50; // [ms] Time between each reading from IMU

int i = 0; // Index for data
float data[200]; // Declaring array that will hold all data points

void setup(){
    Wire.begin();
    Wire.beginTransmission(MPU6050_addr);
    Wire.write(0x6B);
    Wire.endTransmission(true);
    Serial.begin(9600);
    decouple.attach(2); // Assign to pin
    pinion.attach(3); // Assign to pin
    leftshoulder.attach(4); // Assign to pin
    rightshoulder.attach(5); // Assign to pin
    pinMode(inpin1, INPUT_PULLUP); // Pull up to avoid noise activation
    pinion.write(85); // Mid point for continuous servo
    decouple.write(att); // Attach the pinion
    leftshoulder.write(leftend); // Lay down
    rightshoulder.write(rightend); // Lay down
}

void loop(){
    // Button
    int button1 = digitalRead(inpin1); // Read button 1
    int timeT = 0; // For keeping track of how long
    // we are in the IMU reading loop
if (button1 == LOW)  // Assign case
{
    button=1;
    Serial.print("Button_1_
"");
}

// Jump sequence + landing while reading data from IMU
while (button == 1)
{
    pinion.write(67);  // The continuous servo is spinning
    delay(pinionspin);  // Spin during this time
    pinion.write(85);  // Stop the pinion servo

    leftshoulder.write(leftstart);  // Stand up
    rightshoulder.write(rightstart);  // Stand up
    delay(1500);  // Wait 1.5 seconds

    // Read values from IMU
    Wire.beginTransmission(MPU6050_addr);
    Wire.write(0x3B);
    Wire.endTransmission(false);
    Wire.requestFrom(MPU6050_addr, 14, true);
    AccX=Wire.read() << 8 | Wire.read();
    AccY=Wire.read() << 8 | Wire.read();
    AccZ=Wire.read() << 8 | Wire.read();
    GyroX=Wire.read() << 8 | Wire.read();
    GyroY=Wire.read() << 8 | Wire.read();
    GyroZ=Wire.read() << 8 | Wire.read();

    // Print the data in serial monitor
    Serial.println(AccX/16384);
    Serial.println(AccY/16384);
    Serial.println(AccZ/16384);
    Serial.println(GyroX/131);
    Serial.println(GyroY/131);
    Serial.println(GyroZ/131);

    // Detach pinion and rack to jump
    decouple.write(det);  // Release and jump

    // Delay until next reading during jump
    delay(delayT);

    // Repeat
}
APPENDIX C. ARDUINO CODE - IMU DATA

timeT += delayT;

while (timeT < 1200)
{
    // Read values from IMU
    Wire.beginTransmission(MPU6050_addr);
    Wire.write(0x3B);
    Wire.endTransmission(false);
    Wire.requestFrom(MPU6050_addr, 14, true);
    AccX=Wire.read() << 8 | Wire.read();
    AccY=Wire.read() << 8 | Wire.read();
    AccZ=Wire.read() << 8 | Wire.read();
    GyroX=Wire.read() << 8 | Wire.read();
    GyroY=Wire.read() << 8 | Wire.read();
    GyroZ=Wire.read() << 8 | Wire.read();

    // Print the data in serial monitor
    Serial.println(AccX/16384);
    Serial.println(AccY/16384);
    Serial.println(AccZ/16384);
    Serial.println(GyroX/131);
    Serial.println(GyroY/131);
    Serial.println(GyroZ/131);

    // Stretch the legs
    leftshoulder.write(leftend);
    rightshoulder.write(rightend);

    timeT += delayT;  // Delay some time until next IMU reading
}

decouple.write(att);  // Attach the cog
button = 0;  // Exit while loop
delay(1000);

delay(50);  // Refreshrate of the loop
}
Appendix D

All raw IMU data

All data for acceleration and angular velocity during a jump sequence, tested 7 times. The data for every jump is printed in the order of acceleration x-, y- and z-axis, then angular velocity x-, y- and z-axis in a column. This is repeated every 0.1 second for 1.6 seconds.

<table>
<thead>
<tr>
<th>Jump1</th>
<th>Jump2</th>
<th>Jump3</th>
<th>Jump4</th>
<th>Jump5</th>
<th>Jump6</th>
<th>Jump7</th>
</tr>
</thead>
<tbody>
<tr>
<td>−0.8</td>
<td>−0.82</td>
<td>−0.86</td>
<td>−0.86</td>
<td>−0.89</td>
<td>−0.83</td>
<td>−0.82</td>
</tr>
<tr>
<td>−0.02</td>
<td>−0.04</td>
<td>−0.06</td>
<td>−0.03</td>
<td>−7.00</td>
<td>−0.03</td>
<td>−0.04</td>
</tr>
<tr>
<td>0.57</td>
<td>0.53</td>
<td>0.55</td>
<td>0.52</td>
<td>0.49</td>
<td>0.52</td>
<td>0.53</td>
</tr>
<tr>
<td>17.59</td>
<td>17.95</td>
<td>18.56</td>
<td>19.3</td>
<td>18.93</td>
<td>19.42</td>
<td>19.42</td>
</tr>
<tr>
<td>0.3</td>
<td>−0.89</td>
<td>0.35</td>
<td>1.07</td>
<td>1.22</td>
<td>0.29</td>
<td>−0.34</td>
</tr>
<tr>
<td>0.33</td>
<td>1.29</td>
<td>0.84</td>
<td>0.88</td>
<td>−1.25</td>
<td>0.47</td>
<td>0.72</td>
</tr>
<tr>
<td>−0.86</td>
<td>−0.51</td>
<td>−0.87</td>
<td>−0.84</td>
<td>−0.81</td>
<td>−0.77</td>
<td>−0.85</td>
</tr>
<tr>
<td>0.09</td>
<td>−0.03</td>
<td>−0.02</td>
<td>0</td>
<td>−0.09</td>
<td>−0.05</td>
<td>−0.01</td>
</tr>
<tr>
<td>0.46</td>
<td>0.62</td>
<td>0.55</td>
<td>0.48</td>
<td>0.49</td>
<td>0.55</td>
<td>0.43</td>
</tr>
<tr>
<td>17.47</td>
<td>18.44</td>
<td>18.93</td>
<td>19.3</td>
<td>19.3</td>
<td>19.3</td>
<td>19.3</td>
</tr>
<tr>
<td>−3.7</td>
<td>−5.46</td>
<td>−0.74</td>
<td>3.44</td>
<td>4.34</td>
<td>0.37</td>
<td>0.08</td>
</tr>
<tr>
<td>0.92</td>
<td>−2.19</td>
<td>0.8</td>
<td>2.41</td>
<td>2.65</td>
<td>0.58</td>
<td>2.4</td>
</tr>
<tr>
<td>−1.95</td>
<td>0.98</td>
<td>0.15</td>
<td>−0.48</td>
<td>−0.82</td>
<td>−0.06</td>
<td>0.16</td>
</tr>
<tr>
<td>−1.26</td>
<td>0.85</td>
<td>−0.11</td>
<td>−0.4</td>
<td>1.67</td>
<td>−0.27</td>
<td>0.23</td>
</tr>
<tr>
<td>0.78</td>
<td>0.52</td>
<td>−0.05</td>
<td>0.75</td>
<td>−0.26</td>
<td>0.13</td>
<td>−0.46</td>
</tr>
<tr>
<td>17.71</td>
<td>18.44</td>
<td>18.81</td>
<td>19.3</td>
<td>19.05</td>
<td>19.05</td>
<td>19.54</td>
</tr>
<tr>
<td>7.72</td>
<td>−31.44</td>
<td>77.24</td>
<td>84.05</td>
<td>81.37</td>
<td>59.83</td>
<td>45.94</td>
</tr>
<tr>
<td>−149.06</td>
<td>−94.94</td>
<td>−130.37</td>
<td>−140.92</td>
<td>−127.58</td>
<td>−132.66</td>
<td>−102.22</td>
</tr>
<tr>
<td>0.09</td>
<td>−2</td>
<td>−1.24</td>
<td>−0.09</td>
<td>−0.16</td>
<td>0.65</td>
<td>−1.12</td>
</tr>
<tr>
<td>0.05</td>
<td>0.5</td>
<td>1.17</td>
<td>−0.02</td>
<td>−0.12</td>
<td>1.56</td>
<td>2</td>
</tr>
<tr>
<td>−0.26</td>
<td>2</td>
<td>1.34</td>
<td>−0.05</td>
<td>0.18</td>
<td>0.33</td>
<td>2</td>
</tr>
<tr>
<td>17.95</td>
<td>17.95</td>
<td>18.93</td>
<td>19.3</td>
<td>18.93</td>
<td>19.3</td>
<td>19.05</td>
</tr>
<tr>
<td>−39.79</td>
<td>−250.14</td>
<td>−245.55</td>
<td>−48.46</td>
<td>4.13</td>
<td>−167.06</td>
<td>−143.65</td>
</tr>
<tr>
<td>−140.17</td>
<td>−250.14</td>
<td>−207.2</td>
<td>−203.73</td>
<td>−250.14</td>
<td>−250.14</td>
<td>−250.14</td>
</tr>
</tbody>
</table>
### APPENDIX D. ALL RAW IMU DATA

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>X-Axis (m/s²)</th>
<th>Y-Axis (m/s²)</th>
<th>Z-Axis (m/s²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.08</td>
<td>-0.63</td>
<td>0.12</td>
<td>-0.99</td>
</tr>
<tr>
<td>2</td>
<td>0.88</td>
<td>0.38</td>
<td>-0.27</td>
</tr>
<tr>
<td>2</td>
<td>-0.28</td>
<td>-0.66</td>
<td>0.93</td>
</tr>
<tr>
<td>17.71</td>
<td>17.95</td>
<td>19.42</td>
<td>19.3</td>
</tr>
<tr>
<td>-186.24</td>
<td>111.05</td>
<td>53.84</td>
<td>47.69</td>
</tr>
<tr>
<td>250.13</td>
<td>48.9</td>
<td>39.19</td>
<td>40.95</td>
</tr>
<tr>
<td>-0.34</td>
<td>-0.38</td>
<td>0.01</td>
<td>-0.26</td>
</tr>
<tr>
<td>0.13</td>
<td>0.17</td>
<td>-0.27</td>
<td>-0.16</td>
</tr>
<tr>
<td>0.94</td>
<td>1.95</td>
<td>0.74</td>
<td>1.63</td>
</tr>
<tr>
<td>17.10</td>
<td>18.44</td>
<td>18.2</td>
<td>18.2</td>
</tr>
<tr>
<td>124.95</td>
<td>-74.65</td>
<td>33.72</td>
<td>-42.73</td>
</tr>
<tr>
<td>-68.23</td>
<td>-47.98</td>
<td>31.78</td>
<td>7.8</td>
</tr>
<tr>
<td>-0.09</td>
<td>-0.18</td>
<td>-0.01</td>
<td>-0.06</td>
</tr>
<tr>
<td>-0.03</td>
<td>-0.05</td>
<td>0</td>
<td>0.02</td>
</tr>
<tr>
<td>1.18</td>
<td>0.96</td>
<td>1.09</td>
<td>1.16</td>
</tr>
<tr>
<td>17.22</td>
<td>18.2</td>
<td>18.2</td>
<td>19.05</td>
</tr>
<tr>
<td>-28.16</td>
<td>-8.84</td>
<td>-11.31</td>
<td>-7.11</td>
</tr>
<tr>
<td>-7.2</td>
<td>-2.05</td>
<td>-4.95</td>
<td>-2.34</td>
</tr>
<tr>
<td>-0.19</td>
<td>-0.2</td>
<td>-0.15</td>
<td>0</td>
</tr>
<tr>
<td>-0.04</td>
<td>0</td>
<td>0</td>
<td>-0.04</td>
</tr>
<tr>
<td>1.06</td>
<td>1.01</td>
<td>0.95</td>
<td>1</td>
</tr>
<tr>
<td>17.59</td>
<td>17.71</td>
<td>19.05</td>
<td>19.3</td>
</tr>
<tr>
<td>0.52</td>
<td>-2.27</td>
<td>-2.88</td>
<td>0.16</td>
</tr>
<tr>
<td>-0.18</td>
<td>0.24</td>
<td>-0.34</td>
<td>-0.37</td>
</tr>
<tr>
<td>-0.18</td>
<td>-0.15</td>
<td>-0.15</td>
<td>-0.12</td>
</tr>
<tr>
<td>-0.03</td>
<td>-0.01</td>
<td>0.01</td>
<td>-0.02</td>
</tr>
<tr>
<td>1.04</td>
<td>0.98</td>
<td>0.96</td>
<td>0.94</td>
</tr>
<tr>
<td>17.95</td>
<td>17.47</td>
<td>18.56</td>
<td>18.44</td>
</tr>
<tr>
<td>-1.02</td>
<td>-0.64</td>
<td>-2.18</td>
<td>0.44</td>
</tr>
<tr>
<td>0.56</td>
<td>0.7</td>
<td>0.15</td>
<td>0.31</td>
</tr>
<tr>
<td>-0.13</td>
<td>-0.16</td>
<td>-0.13</td>
<td>-0.15</td>
</tr>
<tr>
<td>-0.01</td>
<td>-0.01</td>
<td>0</td>
<td>-0.03</td>
</tr>
<tr>
<td>0.97</td>
<td>0.95</td>
<td>0.96</td>
<td>0.94</td>
</tr>
<tr>
<td>17.95</td>
<td>18.08</td>
<td>18.81</td>
<td>19.05</td>
</tr>
<tr>
<td>0.68</td>
<td>0.76</td>
<td>-0.5</td>
<td>0.31</td>
</tr>
<tr>
<td>1.32</td>
<td>0.89</td>
<td>0.41</td>
<td>1.27</td>
</tr>
<tr>
<td>-0.14</td>
<td>-0.17</td>
<td>-0.15</td>
<td>-0.18</td>
</tr>
<tr>
<td>-0.01</td>
<td>0</td>
<td>-0.02</td>
<td>-0.03</td>
</tr>
<tr>
<td>0.95</td>
<td>0.99</td>
<td>0.97</td>
<td>0.99</td>
</tr>
<tr>
<td>17.10</td>
<td>17.95</td>
<td>18.93</td>
<td>19.05</td>
</tr>
<tr>
<td>0.29</td>
<td>0.75</td>
<td>0.82</td>
<td>-0.09</td>
</tr>
<tr>
<td>0.83</td>
<td>0.69</td>
<td>0.98</td>
<td>0.67</td>
</tr>
<tr>
<td>-0.15</td>
<td>-0.18</td>
<td>-0.18</td>
<td>-0.14</td>
</tr>
</tbody>
</table>


<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-0.02</td>
<td>-0.03</td>
<td>-0.04</td>
<td>0.01</td>
<td>-0.02</td>
</tr>
<tr>
<td>0.94</td>
<td>1</td>
<td>0.99</td>
<td>1</td>
<td>0.99</td>
<td>0.98</td>
</tr>
<tr>
<td>18.08</td>
<td>17.95</td>
<td>19.05</td>
<td>18.81</td>
<td>19.42</td>
<td>19.05</td>
</tr>
<tr>
<td>-0.76</td>
<td>-0.77</td>
<td>0.12</td>
<td>-1.01</td>
<td>-1.16</td>
<td>-10.56</td>
</tr>
<tr>
<td>0.69</td>
<td>0.66</td>
<td>2.13</td>
<td>0.22</td>
<td>-0.11</td>
<td>-1.98</td>
</tr>
<tr>
<td>-0.17</td>
<td>-0.16</td>
<td>-0.19</td>
<td>-0.16</td>
<td>-0.14</td>
<td>-0.16</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-0.03</td>
<td>-0.02</td>
<td>-0.01</td>
<td>-0.02</td>
</tr>
<tr>
<td>0.98</td>
<td>0.97</td>
<td>0.95</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>17.95</td>
<td>18.08</td>
<td>18.93</td>
<td>18.93</td>
<td>19.54</td>
<td>18.93</td>
</tr>
<tr>
<td>-1.98</td>
<td>-0.05</td>
<td>-0.85</td>
<td>-0.98</td>
<td>-2.39</td>
<td>-0.44</td>
</tr>
<tr>
<td>-0.33</td>
<td>0.75</td>
<td>-1.55</td>
<td>0.26</td>
<td>0.5</td>
<td>1.15</td>
</tr>
<tr>
<td>-0.16</td>
<td>-0.15</td>
<td>-0.18</td>
<td>-0.14</td>
<td>-0.15</td>
<td>-0.15</td>
</tr>
<tr>
<td>-0.03</td>
<td>-0.02</td>
<td>-0.01</td>
<td>-0.01</td>
<td>-0.01</td>
<td>-0.02</td>
</tr>
<tr>
<td>1</td>
<td>0.99</td>
<td>0.97</td>
<td>0.96</td>
<td>0.97</td>
<td>0.97</td>
</tr>
<tr>
<td>17.59</td>
<td>17.95</td>
<td>18.44</td>
<td>18.44</td>
<td>18.56</td>
<td>18.81</td>
</tr>
<tr>
<td>0.21</td>
<td>0.44</td>
<td>-6.94</td>
<td>0.37</td>
<td>0.02</td>
<td>0.18</td>
</tr>
<tr>
<td>0.74</td>
<td>1.35</td>
<td>-2.33</td>
<td>1.01</td>
<td>0.6</td>
<td>0.98</td>
</tr>
<tr>
<td>-0.17</td>
<td>-0.15</td>
<td>-0.15</td>
<td>-0.14</td>
<td>-0.18</td>
<td>-0.15</td>
</tr>
<tr>
<td>-0.03</td>
<td>-0.03</td>
<td>-0.02</td>
<td>0</td>
<td>-0.04</td>
<td>-0.02</td>
</tr>
<tr>
<td>1</td>
<td>0.99</td>
<td>0.98</td>
<td>0.98</td>
<td>1.01</td>
<td>0.97</td>
</tr>
<tr>
<td>17.95</td>
<td>17.95</td>
<td>18.93</td>
<td>19.05</td>
<td>19.54</td>
<td>19.05</td>
</tr>
<tr>
<td>0.58</td>
<td>0.08</td>
<td>0.76</td>
<td>1.23</td>
<td>-0.29</td>
<td>0.17</td>
</tr>
<tr>
<td>1.08</td>
<td>1.01</td>
<td>0.75</td>
<td>1.33</td>
<td>0.41</td>
<td>0.91</td>
</tr>
<tr>
<td>-0.16</td>
<td>-0.18</td>
<td>-0.15</td>
<td>-0.17</td>
<td>-0.16</td>
<td>-0.15</td>
</tr>
<tr>
<td>-0.01</td>
<td>-0.02</td>
<td>-0.01</td>
<td>-0.02</td>
<td>-0.02</td>
<td>-0.02</td>
</tr>
<tr>
<td>0.98</td>
<td>0.99</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
<td>0.99</td>
</tr>
<tr>
<td>17.95</td>
<td>17.71</td>
<td>18.44</td>
<td>19.05</td>
<td>18.56</td>
<td>19.3</td>
</tr>
<tr>
<td>-0.6</td>
<td>-0.25</td>
<td>0.09</td>
<td>1.39</td>
<td>0.37</td>
<td>-0.13</td>
</tr>
<tr>
<td>0.58</td>
<td>0.23</td>
<td>0.99</td>
<td>0.65</td>
<td>0.73</td>
<td>0.51</td>
</tr>
</tbody>
</table>