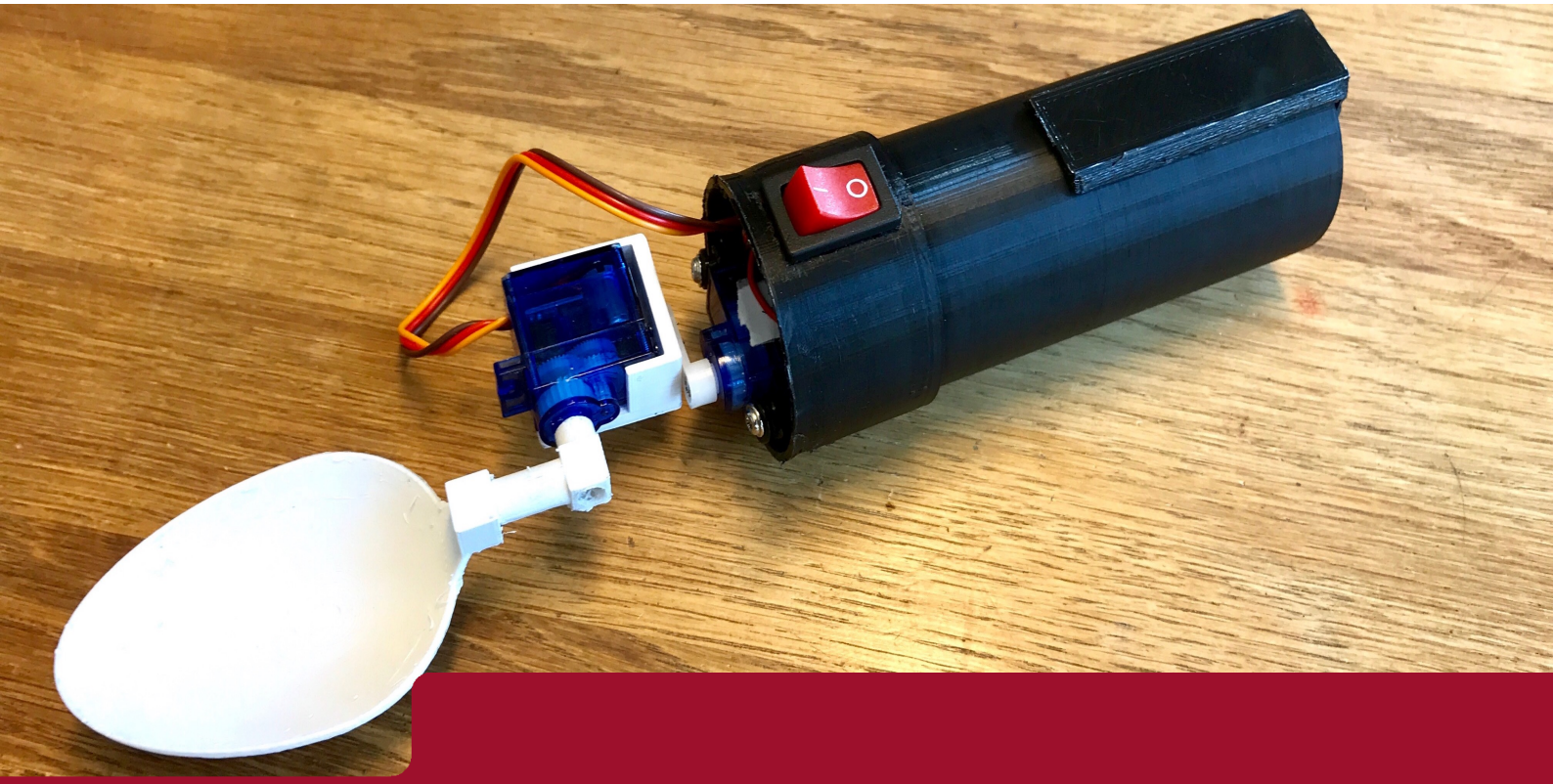
DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, *SWEDEN 2017*

The Stabilizing Spoon

Self-stabilizing utensil to help people with impaired motor skills

JOHAN ABRAHAMSSON

JOHAN DANMO



KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF INDUSTRIAL ENGINEERING AND MANAGEMENT

The Stabilizing Spoon

Self-stabilizing utensil to help people with impaired motor skills

JOHAN ABRAHAMSSON
JOHAN DANMO

Bachelor's Thesis at ITM
Supervisor: Baha Alhaj Hasan
Examiner: Nihad Subasic

TRITA MMK 2017-21 MDAB 639

Abstract

The technology for assisting people who are functionally challenged has improved over the recent decades. With today's technology, people with Parkinson's disease can, with a device on their wrist, be able to draw pictures. Human limbs lost due to accidents can be replaced with bionic limbs and with help from smartphones, blind people can by audio be informed what kind of object that appear in front of them. These are a few examples where technology has eased everyday life for people with impaired functionality.

The purpose of this thesis is to analyze how an Arduino microcontroller can be utilized to help people with impaired motor skills during their eating process. A prototype of a stabilizing spoon was constructed to work under real circumstances and intended to be a complement for people who are in need of assistance during their eating process.

To make this possible, a sensor with gyroscopes combined with accelerometers was used to identify which direction the device's handle was being tilted, as well as how fast its position was changed. Two servo motors were placed orthogonally to each other to establish a system of two degrees of freedom. With this setup, the spoon was intended to maintain its spoon bowl in a horizontal position. Experimental results of the spoon showed promising performance with some limitations.

Referat

Den stabiliserande skeden

Teknologiska hjälpmedel för människor som har en funtionsnedsättning har förbättrats under de senaste decennierna. Med dagens teknologi kan personer drabbade av Parkinson's disease, med en apparat på deras handled, vara kapabla till att måla teckningar. Mänskliga lemmar som förlorats vid olyckor kan ersättas med bioniska lemmar, och med hjälp av smartphones kan blinda personer bli informerade om vilken typ av objekt som finns framför dem. Detta är bara några få exempel på när teknologi har underlättat vardagen för människor med funktionsnedsättning.

Syftet med detta projekt är att undersöka hur en Arduino mikrokontroller kan implementeras vid måltider för att assistera människor som har en motorisk funktionsnedsättning. En prototyp av en stabiliserande sked konstruerades för att fungera under verkliga förhållanden och dess intention var att verka som ett komplement för människor som behöver assistans vid måltider.

För att göra detta möjligt användes en sensor med gyroskop och accelerometer för att identifiera vilken riktning enhetens handtag var lutad åt samt hur snabbt handtaget förflyttades. Två servomotorer placerades ortogonalt mot varandra för att skapa ett system med två frihetsgrader. Med denna uppsättning var det tänkt att skeden skulle hålla dess skedskål i en horisontell position. Experimentella resultat av skeden visade lovande prestanda med några begränsningar.

Acknowledgements

We would like to thank Baha Alhaj Hasan for supervising us through the project and leading us towards our goal as well as giving us advice when we needed it. We would also like to thank the assistants, Fredrika Kringberg, Marcus Olsson, Philip Pulli and Simon Gärtner who were involved in the course and helped when present in the laboratory, regardless if scheduled as an assistant or not. Our laboratory engineer, Staffan Qvarnström added great value to the project in terms of sharing knowledge of components as well as ordering them. Special thanks goes to all our fellow students who helped us, but especially Bohan Zhou and Karl Enoksson who shared and discussed ideas with us regarding the report and the project in general.

Johan Danmo and Johan Abrahamsson
Stockholm, May, 2017

Contents

List of Figures

List of Tables

Nomenclature

Abbreviation

$3D$	Three-Dimensional
CAD	Computer Aided Design
CPU	Central Processing Unit
DC	Direct Current
DMP	Digital Motion Processor
GND	Electrical Ground
GPS	Global Positioning System
IDE	Integrated Developed Environment
IMU	Inertial Measurement Unit
KTH	Kungliga Tekniska Högskolan/Royal Institute of Technology
$MEMS$	Micro Electrical Mechanical Systems
PD	Parkinson's Disease
PI	Proportonal Integral part
PLA	Polyactic Acid
PWM	Pulse Width Module
USB	Universial Serial Bus

Symbols

α	Degree symbol alpha
$\hat{z}$	Filtered signal

G(s)	Low pass filter
x	Low frequency noise
y	High frequency noise

Chapter 1

Introduction

1.1 Background

The use of stabilizing mechanisms appear in many different fields. A few examples can be found in air crafts, industrial robots and camera stabilization [camfere.com, 2016]. Some of these systems have a desire for a steady and non-changeable position of a body, regardless of involved motion. This notion is applicable in assisting people with tremors. With help from tremor compensating devices, people who are affected by this disability can execute some tasks that occur in everyday life. With a stabilizing spoon, a person with these issues, can eat independently and does not need a helping hand from another. A group that suffer from this ailment are people with Parkinson's disease. Parkinson's disease (PD) is a neurological degenerative disease that causes uncontrollable shaking and makes it difficult for the affected person to eat. There is no cure for PD, but there is technology and potential for new technology that can help people who carry the disease with their daily lives.

The aim of this project is to produce a stabilizing spoon that will compensate for unintended motions, such as tremors. With a low budget, the goal is to make a highly efficient prototype that consists mainly of a microcontroller and servo motors. There are several different technological assisting spoons on the market today, but the products are unfortunately quite expensive [liftware.com, 2016]. With results from this project, further research could be able to continue developing cheaper and better stabilizing spoons.

1.2 Purpose

The purpose of this report is to investigate how an Arduino microcontroller can be implemented to help people with impaired motor skills. The research question to be answered is:—

"How can an Arduino microcontroller be utilized to help people with impaired

motor skills during their eating process?"

To answer this research question, two sub-questions that are related to the main research question were formulated. Their aim is to be more specific and delimit the research field of the project and state as the following:—

"How fast can the device react for a motion and will it be fast enough to help people with high frequency tremors?"

"To what extent will the device fit the hand of the user?"

1.3 Scope

The spoon is intended to have two degrees of freedom, rotational movement around the x-axis, roll, and rotational movement around the y-axis, pitch, for more illustration see Figure 1.1. The z-axis which is orthogonal both to the x- and y-axis, will have no motor for counteracting motion since two degrees of freedom will be satisfying enough for this project. To consider the second sub-research-question, "To what extent will the device fit the hand of the user?", the device can not be too large.

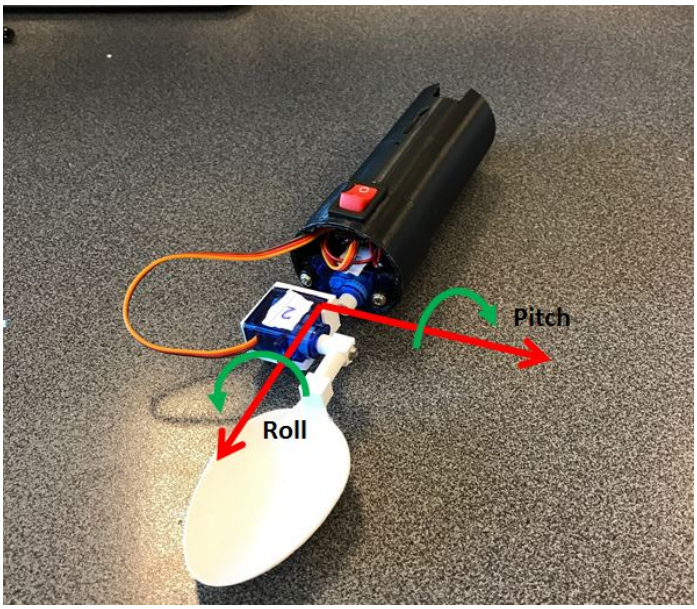


Figure 1.1. The Stabilizing Spoon with pitch- and roll-axis

1.4 Method

The prototypes were constructed with machines and tools available at the KTH Industrial Technology and Management department. The essential machines and tools were 3D-printers, soldering tools, drills and screwdrivers. All the 3D-printed parts were designed in the CAD-software program Solid Edge ST8 and printed by an Ultimaker 2. All programming regarding the behavior of the spoon was done in Arduino's Integrated Developed Environment (IDE). One of the first prototypes were constructed with styrofoam containing all the electronics and servos, then several iterations of the construction were made of 3D-printed parts made out of PLA hard plastic. To test the performance of the spoon and see its behavior through graphs, the numerical programing software Matlab (version 2016a) was used.

1.5 Related Projects

Stabilizing spoons exist on the market today and there are some projects that are fairly similar to this project. Several of the related projects that are stabilizing spoons use servo motors and IMUs as it is intended in this project.

One similar project is shown below in Figure 1.2, where the motion of the spoon is controlled through a PI-regulator [Khazane et al., 2015] with a complementary filter to receive proper values from the IMU [Van de Maele, 2013].

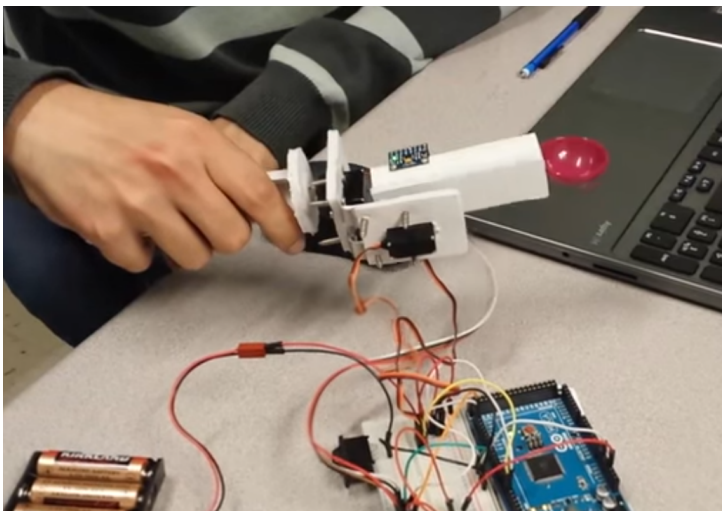


Figure 1.2. A Team 4 Report [Khazane et al., 2015]

Another project, see Figure 1.3, is using the same kind of IMU but uses a Kalman filter instead of a complementary filter to get smoother values.

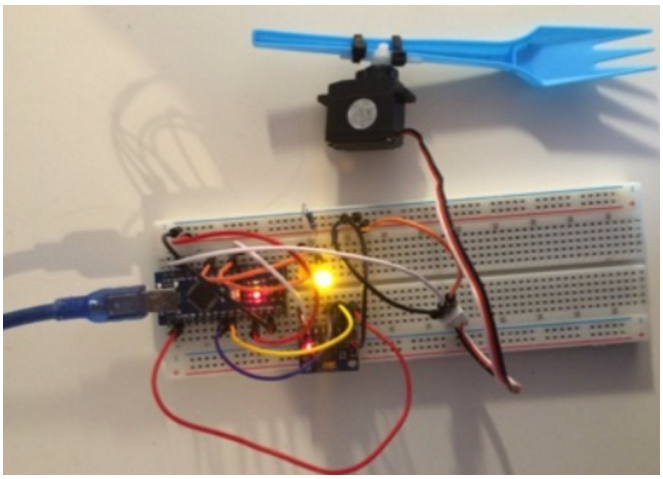


Figure 1.3. Third year mechanical engineering project [Birtwell, 2016]

Chapter 2

Theory

2.1 Stabilizing spoons

A stabilizing spoon is a device which maintains a horizontal position of its front regardless of the motion it receives from the user at the rear end of the spoon. For a theoretical demonstration of the spoon's movement, see Figure 2.1. Stabilizing spoons have been developed foremost to help people with tremors and people who are functionally challenged with difficulties moving their hands. Its aim is to assist these people so they can eat independently. People with hand tremors could be persons with Parkinson's disease and people who are functionally challenged could be persons with Cerebral palsy.

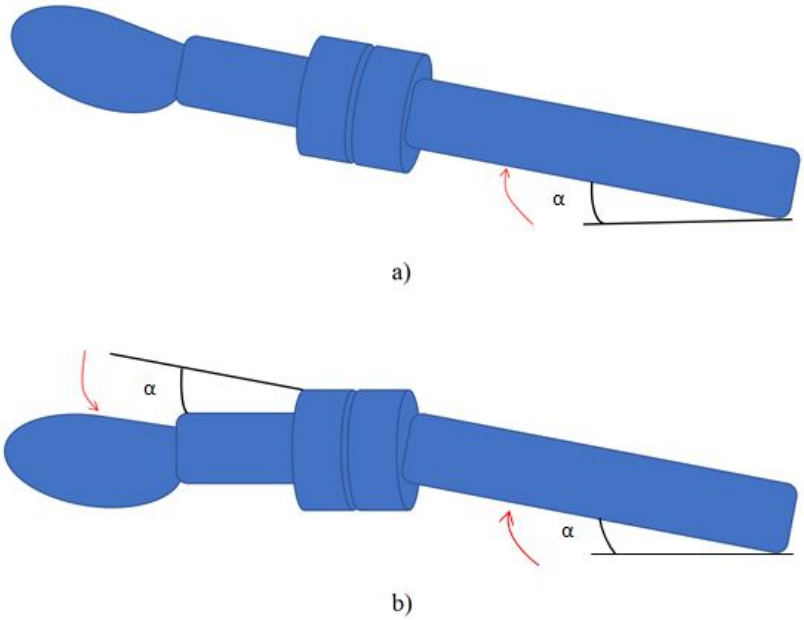


Figure 2.1. Spoon's movement illustrated
a) without compensation
b) with compensation

If the handle of the spoon is tilting an angle with α degrees, see Figure 2.1, actuators in the spoon's construction will compensate with the same angle α and put the spoonbowl in its initial horizontal position.

2.2 Inertial Measurement Unit

An Inertial Measurement Unit (IMU) is an electronic device that can be found in aircrafts, GPS-systems and satellites. It measures and reports the specific acceleration and angular rate of a body. The IMU consists of a combination of three gyroscopes and three accelerometers placed orthogonally to each other forming a coordinate system [invensense.com, 2017]. An accelerometer measures inertial acceleration and a gyroscope measures rotational position in reference to an arbitrary chosen coordinate system. Combined they can detect where an object is in space and if its tilting.

2.3 Servo motor

Servo motors are commonly used in hobby projects such as building robots. A servo motor has a very high precision in position feedback, i.e. it can tell with a high accuracy how many degrees the motor shaft has rotated. It consists of a motor that is paired with a form of a potentiometer (pot) and a servo control circuit, see Figure 2.2. The potentiometer is connected to the output shaft and is also proportional to it which means that the potentiometer register every degree the shaft has rotated. With this setup, a closed loop feedback system is established within the servo. The loop does not include the microcontroller which results in making the microcontroller just send commands to the servo but not receive any information from the servo.

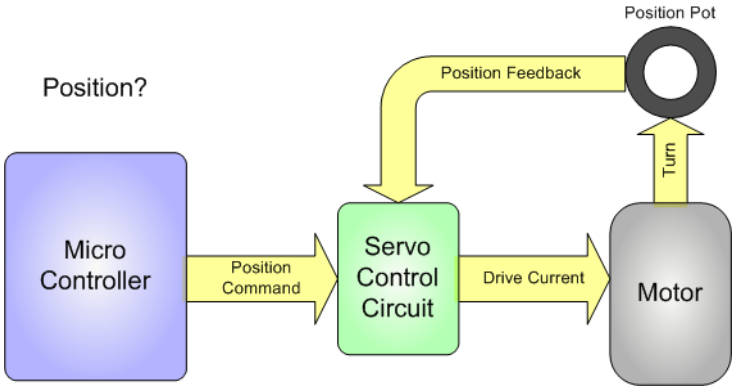


Figure 2.2. Schematic over a feedback servo motor [learn.adafruit.com, 2017]

2.4 Microcontroller

A microcontroller can be considered as a small computer that can be programmed to control electronics. It consists of one or more CPUs along with memory and programmable inputs/outputs. It has a broad area of use and can be found in automobile engine control systems, medical devices, toys and remote controls. The largest names within microcontrollers in hobby projects are Arduino and Raspberry Pi. In short, Raspberry Pi has a large processing power and is used for a wide range of applications, the Arduino comparatively, tends to be used for projects with less processing power [Orsini, 2014].

2.5 Quaternions

Quaternions is a mathematical method which is useful when describing motions in 3D. The theory is commonly used in computer graphics, for example when an object in the graphics makes a transition from a rotation to another. Unlike Euler angles,

using quaternions will avoid the problem of a locking position and is therefore simpler to compose.

A quaternion can be seen as a vector in a 3-dimensional Cartesian coordinate system, the vector or quaternion can point in any direction in this coordinate system and rotate around its own axis, see Figure 2.3. This will allow the quaternion, or the object that is being manipulated, to rotate in any direction without being at risk for a locking position. A locking position, or ”gimbal lock” as it is called, is a loss of one degree of freedom in a three dimensional, three gimbal mechanism. This phenomenon ”locks” the system into rotating in a degenerate two dimensional space and allows the object to rotate only in two directions instead of three [madsci.org, 1998]. It occurs when the manipulated object is rotating making two or three gimbal axes align and become parallel to each other, see Figure 2.4.

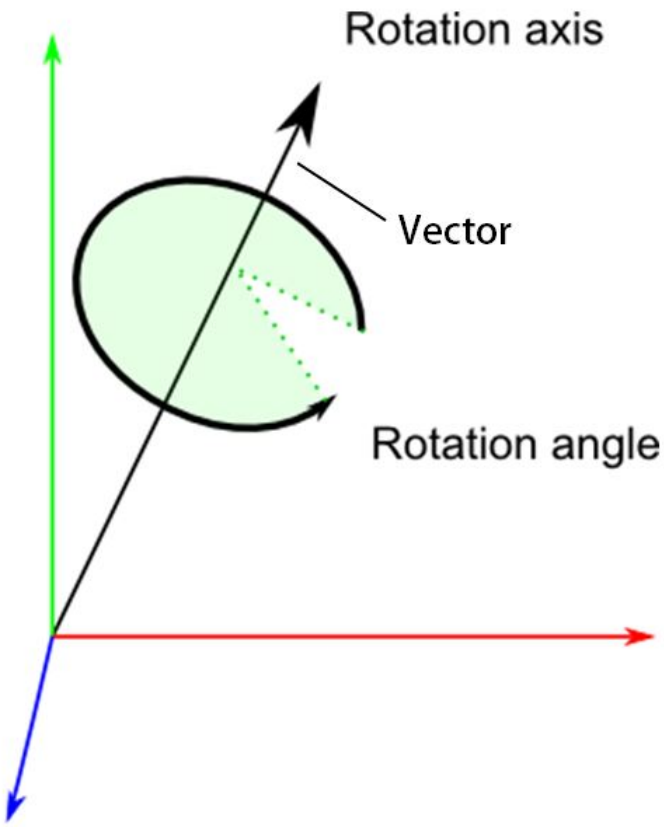


Figure 2.3. Vector in 3D-space [opengl tutorial.org, 2017]

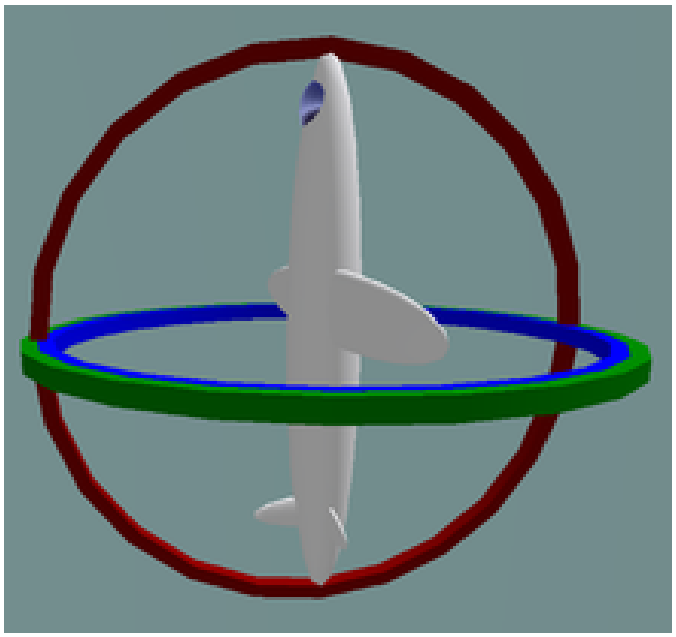


Figure 2.4. Gimbal lock problem where the x- and y-gimbal axis are aligned making the object lose one degree of freedom [MathsPoetry, 2009]

2.6 PWM - Pulse Width Modulation

To control electronics through a microcontroller, PWM is commonly used. PWM signals are square waves, switching on and off with a certain voltage. The unit that is being controlled by the PWM signals behaves accordingly after each duty cycle. A duty cycle is the percentage of time when the voltage is ”on” which means that if a motor is being controlled, the duty cycle will determine indirectly its velocity. The example of the 100% duty cycle, which is shown in Figure 2.5, will result in transferring power continuously to the unit that is being controlled, which in the case of the motor will result in outputting its maximum velocity [arduino.cc, 2016b].

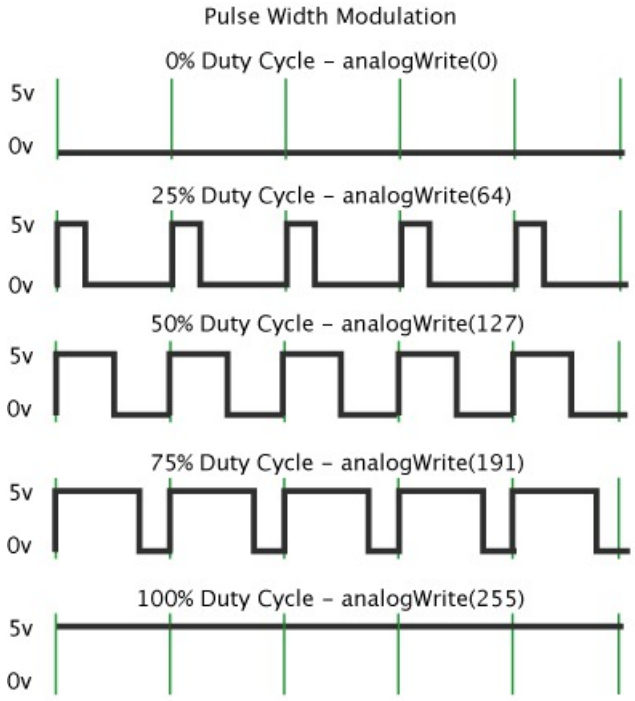


Figure 2.5. PWM signals with various duty cycles [arduino.cc, 2016a]

2.7 Complementary filter

An IMU presents occasionally values that are incoherent and vary unreasonably to one another. To get proper values, i.e. clear values, a filter is needed. A complementary filter consists of one low pass filter and one high pass filter. Figure 2.6 shows a block diagram where y is a high frequency signal and x is a low frequency signal going though a low pass filter, G(s), respectively a high pass filter, 1-G(s), which then results in a filtered signal, $\hat{z}$ [Higgins, 1975].

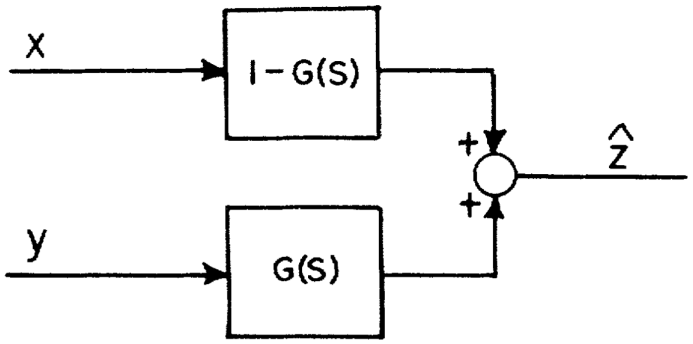


Figure 2.6. Complementary filter [Higgins, 1975]

The implementation of this setup of a complementary filter in software is shown in the equation

$$\hat{z} = x(1 - G(S)) + yG(S) \tag{2.1}$$

where the fraction, G(S) is generally 0.98 and the parameters x and y represent values from the accelerometer respectively the gyroscope from the IMU.

Chapter 3

Demonstrator

3.1 Problem formulation

The demonstrator was created to enable testing and evaluate its usability. The problems that had to be solved was the following;

1. Counteracting of motions

2. Dimensions of the device to fit the user's hand

Two different codes have been used separately for the counteracting of motions. One is developed by the members of this project and the other one has been used from another stabilizing project [Rowberg and Poynter, 2016]. Both of the codes have been using the output values from the IMU and then transforming them into rotational commands for the servo motors, for a simple overview see Figure 3.1. The essential difference between the codes concerning the counteracting is how the values from the IMU have been transformed into rotational commands. The two codes differ in their calculations and algorithms for computing the angles, they also differ in using a filter, the project's own developed code is using a filter while the other one is not.

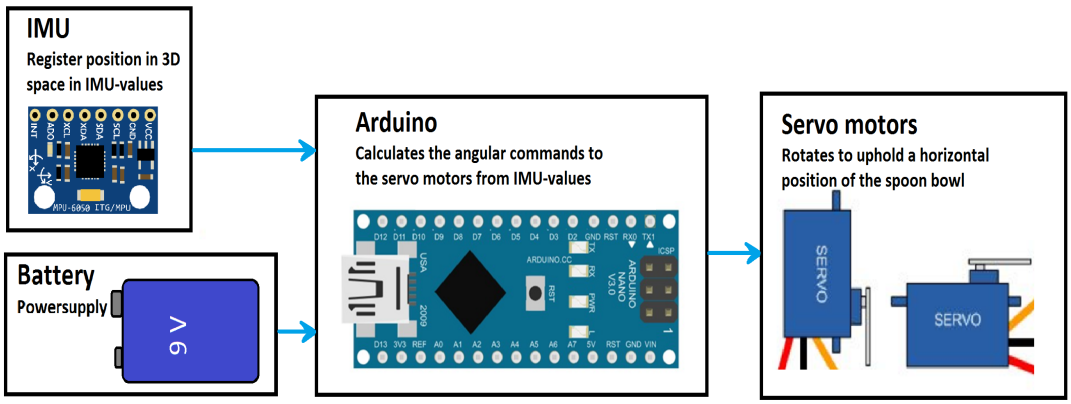


Figure 3.1. Overview of the involving components, created in Paint

The size of the product had to be taken into consideration since the device, as stated above, must fit into the users hand. During the project's time three prototypes were made. In Prototype one, left in Figure 3.2, the components were placed in a way that everything could be observed. Prototype two, right in Figure 3.2, had two levels in its casing, one for the circuit board and one for the servomotor and battery. The following and last prototype (Prototype 3) was made with a handle that was cylindrical to make the device more fit to the hand of the user, see Figure 3.3.

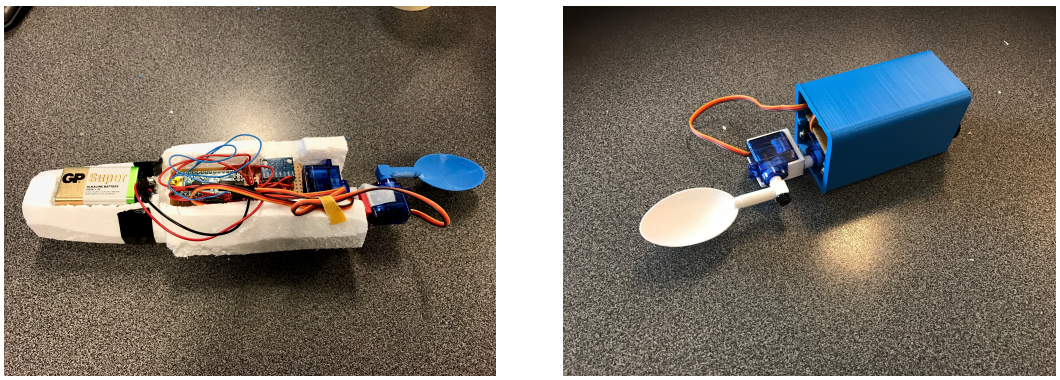


Figure 3.2. Prototype one and two

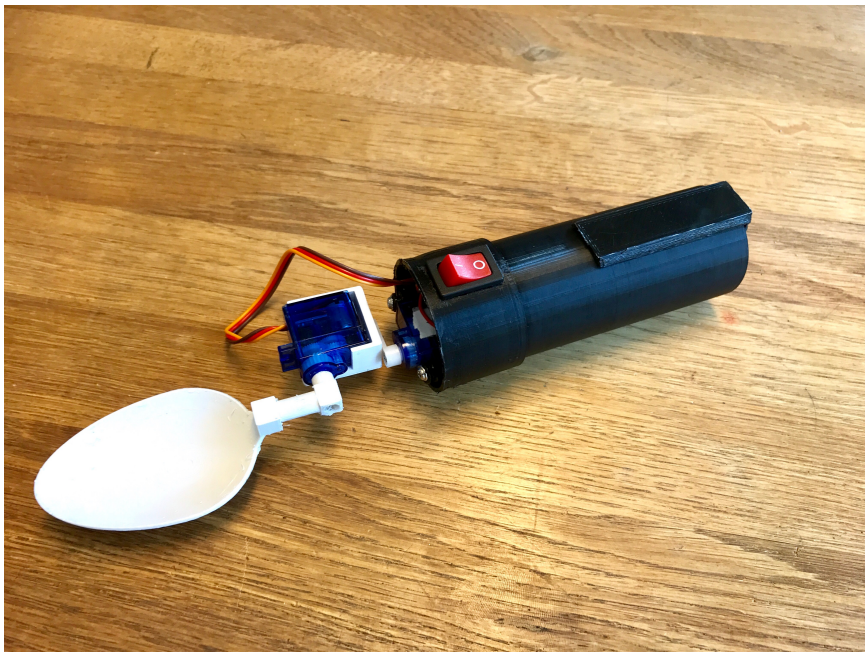


Figure 3.3. Prototype three

3.2 Electronics

The electrical components used in this project were connected with the Arduino Nano as demonstrated in the following electronic schematic, Figure 3.4.

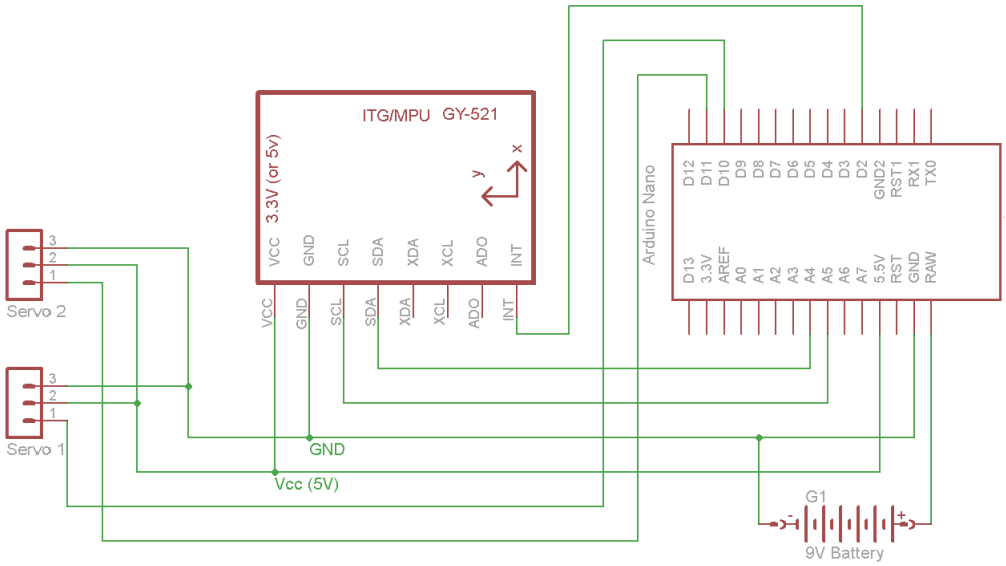


Figure 3.4. Schematic coupling of electronics, created in EAGLE

3.2.1 Microcontroller - Arduino Nano

The Arduino Nano microcontroller is a small device and therefore suitable for small projects. The dimension of the board is 45 x 17 mm and approximately 10 mm thick with the pins attached. It has about the same specifications as the Arduino Uno [arduino.cc, 2017b] but lacks a DC power jack. It is a breadboard based on the microchip ATmega 328 and equipped with eight analog pins and 14 digital pins, of which six are PWM enabled [arduino.cc, 2017a]. The mini-USB port is used for programming but can also be used as a power source. All Arduino boards have software included, Arduino IDE. This means the board can be plugged into a computer and be programmed through Arduino's own software. Arduinos software is an Open-source code which is based on C/C++ [opensource.com, 2017].

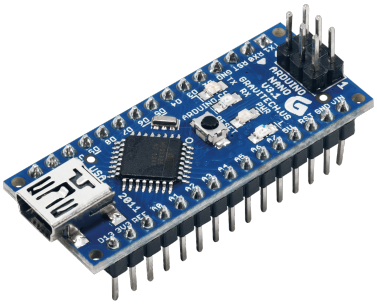


Figure 3.5. The Arduino Nano [electronics lab.com, 2017]

3.2.2 Servo motor - Micro 9g servo FS90

Micro 9g servo FS90 is a small yet powerful servo motor with the ability to rotate approximately 180 degrees with a torque output of approximately 0,127 Nm. The dimension of the servo motor is 23.2 x 12.5 x 22.0 mm and weighs 9 grams as the name states. The motor have 3 pins in total, one for the signals (PWM), one for the input voltage (Vcc) with an operating voltage of about 4.8 V or 6 V and one for electrical ground (GND) [MantechElectronics, 2017]

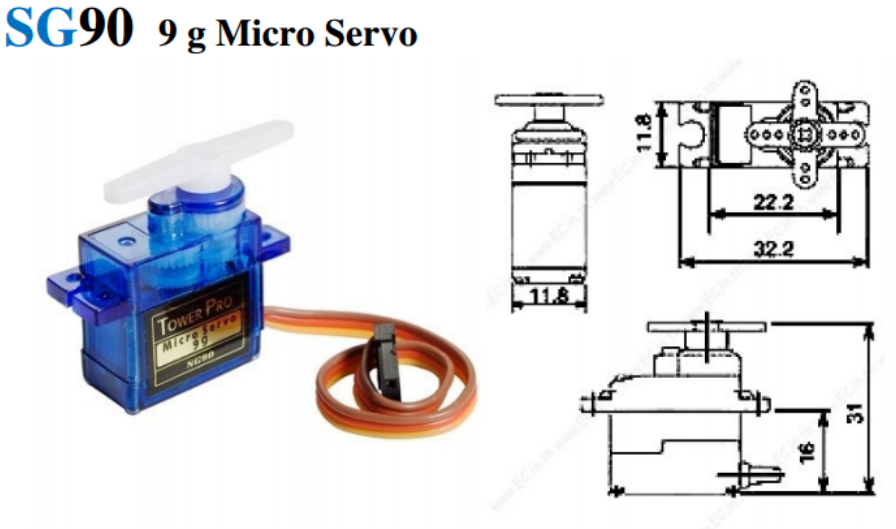


Figure 3.6. TowerPro SG90 Mini Micro Digital Servo 9g and its dimensions [micropik.com, 2014]

3.2.3 IMU - MPU6050

The MPU6050 combines a three-axis gyroscope and a three-axis accelerometer on the same board, along with an onboard Digital Motion Processor™(DMP™) it can accurately track user motions. With the gyroscope and the accelerometer, which both are triple axis Micro Electrical Mechanical Systems (MEMS), it can provide information of the boards angular position and acceleration [invensense.com, 2017]. Due to its size and the easy accessible Open-source libraries, the MPU6050 was ideal for the project.

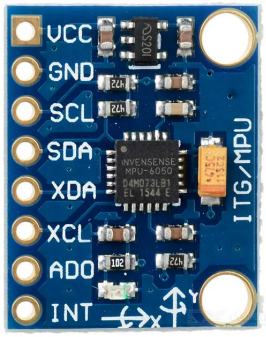


Figure 3.7. The MPU6050 [dx.com, 2017]

3.3 Software

Two different types of codes have been used separately in this project and both will be discussed in the following sub-sections.

3.3.1 The project's own developed code

The code that was developed by the project's members has essentially three parts. The first part is bound to the declaration of each parameter used later in the code. The second part is the initial setup, establishing communication between the microcontroller, servomotor and the IMU. The third part is retrieving the value outputs from the IMU, which it later transforms into rotational commands for the servo motors through a complementary filter.

Figure 3.8 demonstrates the work flow through a flowchart, from the start of the code to the rotational commands of the motors.

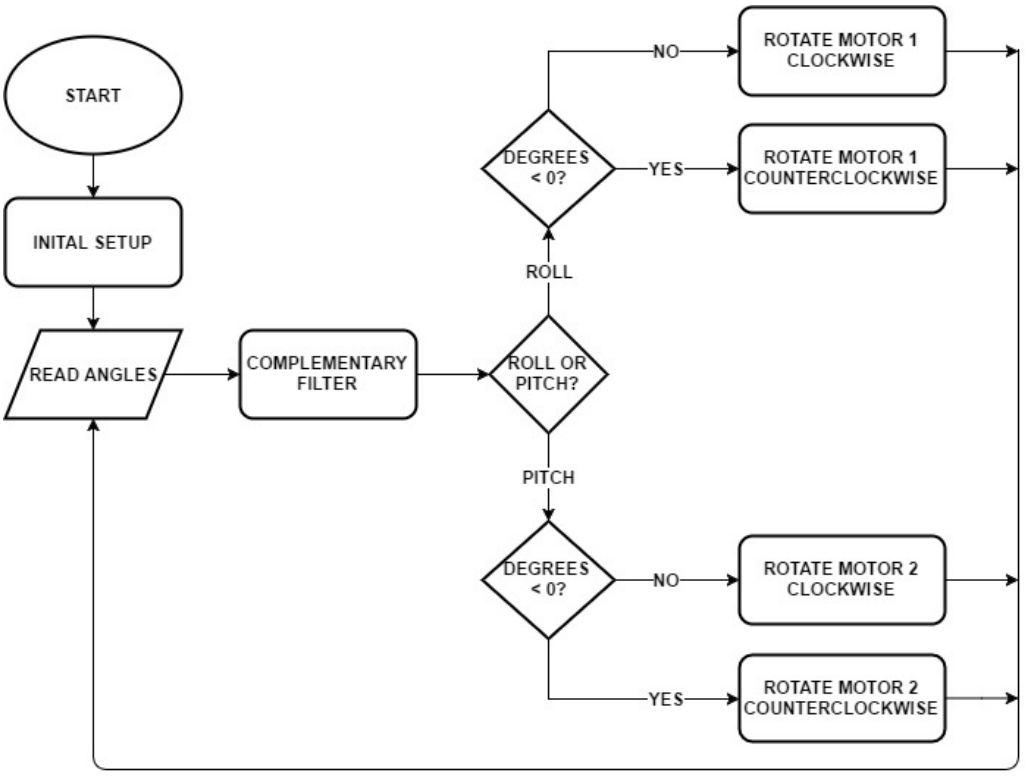


Figure 3.8. Flowchart over the project's own developed code, drawn in the free software program Draw.io

3.3.2 Code from a similar stabilizing project

This code is also divided into three parts like the other code. The first and the second part is roughly the same as in the other code. The main difference is the third part where the IMU values are transformed into rotational commands. Instead of letting a complementary filter refine the IMU values and then transforming them into rotational commands, the transformation is done by Quaternions.

3.4 Hardware

All of the mounting parts and the casing were made out of PLA (hard plastic) constructed by an Ultimaker 2 3D printer.

The casing of Prototype 3 is shown below in Figure 3.9 which is a hollow cylinder with details for mounting. One of the servo motors was mounted with screws in the holes shown in the figure while the circuit board was placed inside the cylinder on a rack above the battery.

Figure 3.9. CAD-model of the Casing

The other servo motor was mounted to the first mentioned servo motor using a standard servo rotor connected with screws to a simple container, Figure 3.10, which it was held in.

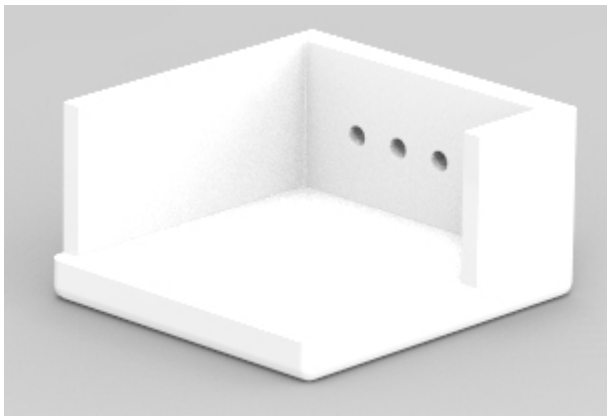


Figure 3.10. CAD-model of the Container

The last hardware component is the spoon bowl, Figure 3.11, which was screwed directly onto the outermost servo motor's rotor.

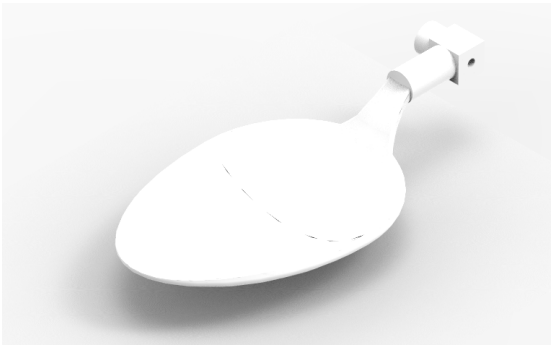


Figure 3.11. CAD-model of the spoon shaft

3.5 Results

In this section the performance of the spoon will be presented. The section will be divided into two parts where one is covering the results from the code that has been used from the similar stabilizing project and the other part is covering the results from the project's own developed code. The test rig used to gather numerical results of the reaction time can be seen in Figure 3.12 where two Arduino Uno were connected with two IMU6050 separately to gather the data. The data was then used in Matlab [mathworks.com, 2017] to create graphs to visualize the behavior of the spoon.

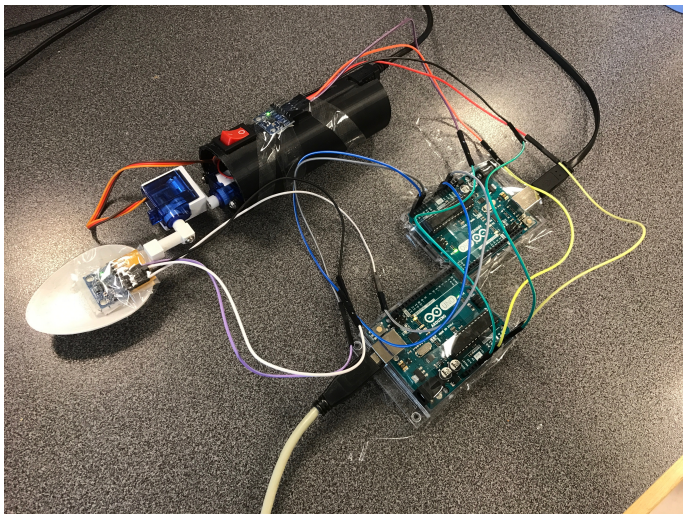


Figure 3.12. Test rig of the device with two Arduino Uno and two IMU6050

3.5.1 Code from a similar stabilizing project

The following graphs in Figure 3.13 shows the pitch- and roll movement of the spoon when tilted slowly.

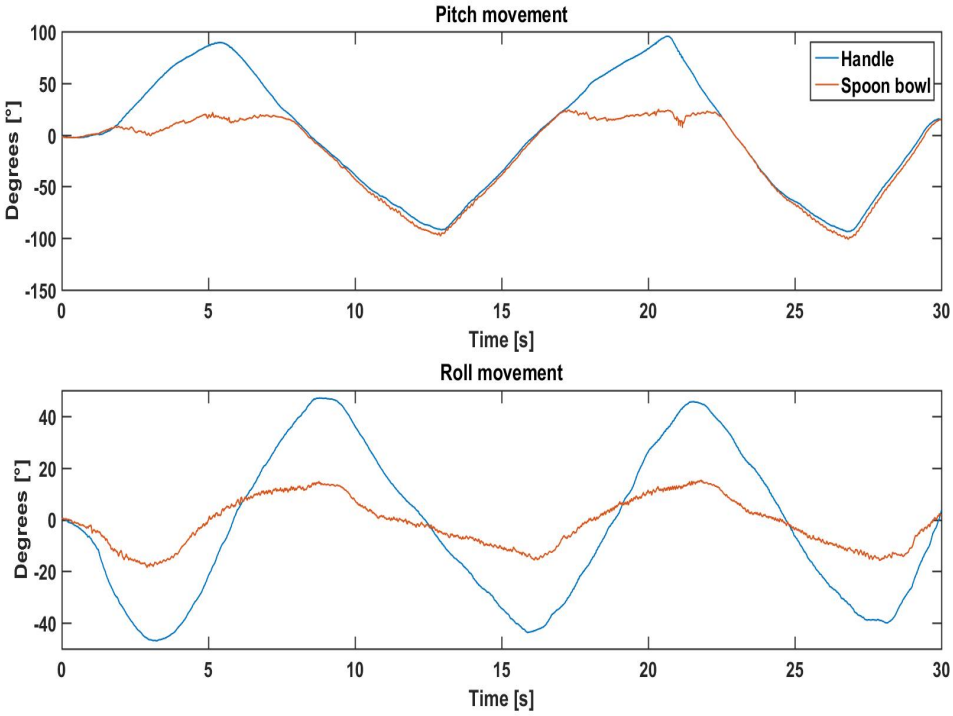


Figure 3.13. Performance of the spoon's pitch and roll movement in slow oscillation, created in Matlab

When the handle is undergoing a negative pitch motion in the upper graph of Figure 3.13, i.e. the handle is tilting degrees below zero (vertical axis), both curves are coinciding. This phenomenon occurs because the spoon bowl is following the direction of the handle to easily pick up objects. In other words, instead of the spoon bowl remaining in its horizontal position while moving the rear end of the handle upwards, the spoon bowl will follow the handle and the user will be able to shovel up the desired object, see Figure 3.14.

Figure 3.14. Spoon held in a negative pitch motion, i.e. rear-upward motion

Looking at the graph of the pitch movement in Figure 3.13, one can see that when the handle is tilting 90°, the spoon bowl is stabilized around 20°. When undergoing roll movements, which aslo can be seen Figure 3.13, the spoon bowl is turning from -15° to 15° when the handle is rolling from -45° to 45°.

To simulate hand tremors, the device was tilted more rapidly. The results are shown in Figure 3.15. As a positive pitch motion is applied to the spoon (upper graph), i.e. a rear-downward motion, see Figure 3.16, the spoon bowl has a tendency to follow the handle to about 25° until it stabilizes to around 10°. When the device is experiencing a roll motion (lower graph), the spoon bowl follows the motion of the handle until it stabilizes quite fast at ±25°.

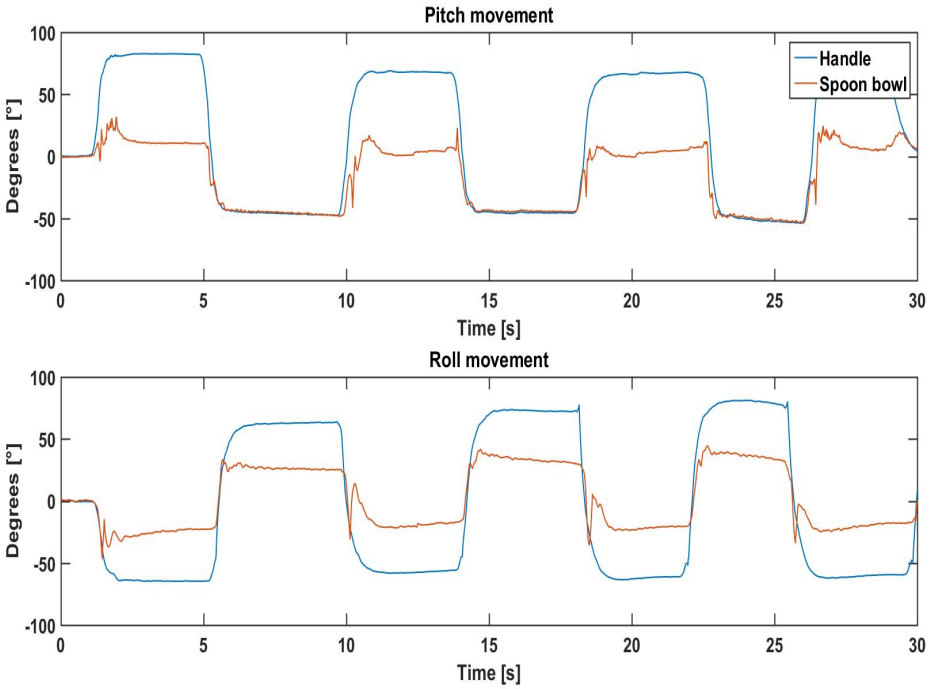


Figure 3.15. Performance of the spoon's pitch and roll movement in fast oscillation, created in Matlab

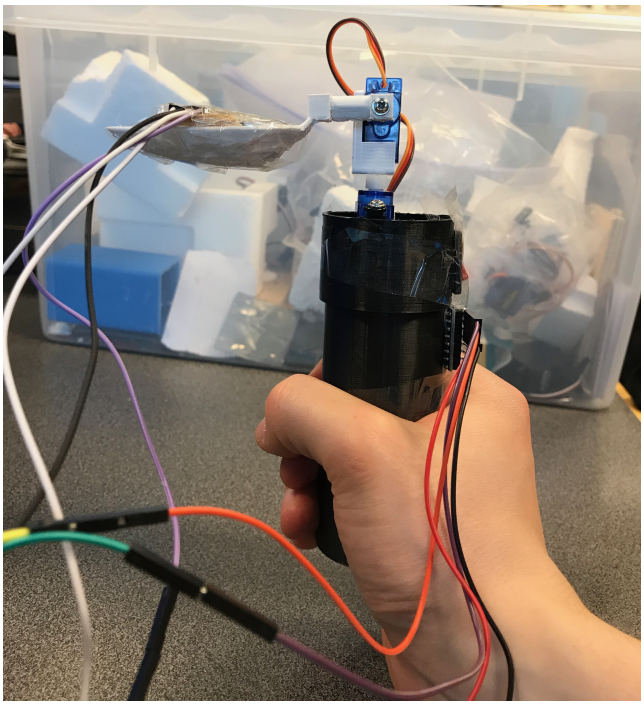


Figure 3.16. Spoon held in a positive pitch motion, i.e. rear-downward motion

3.5.2 The project's own developed code

The results from the project's own developed code are quite similar to the results from the code that has been taken from the other project, especially for the roll movements. Figure 3.17 shows how the device reacts when the same motions are applied to the device as in section 3.5.1. The difference is mainly the pitch motion (upper graph) where there is a peak in each period (around the 5" mark and the 21" mark). This occurs because the servo motor has reached its end point. It reaches its end point due to a displacement of the rotation span which is caused by the different coordinate system setup of the IMU, compared to the other code.

The deviation for the spoon bowl in the roll motion (lower graph) during the period of disturbance is quite equivalent to the other code if comparing Figure 3.13 and Figure 3.17. The difference though is that the spoon bowl rather overcompensates than follow the curve of the handle as it does with the other code.
When the device experiences rapid disturbances as shown in Figure 3.18, the spoon bowl follows the movement of the handle to about 50° in the pitch motion and ±50° in the roll motion, then it stabilizes at around 10° respectively 20°.

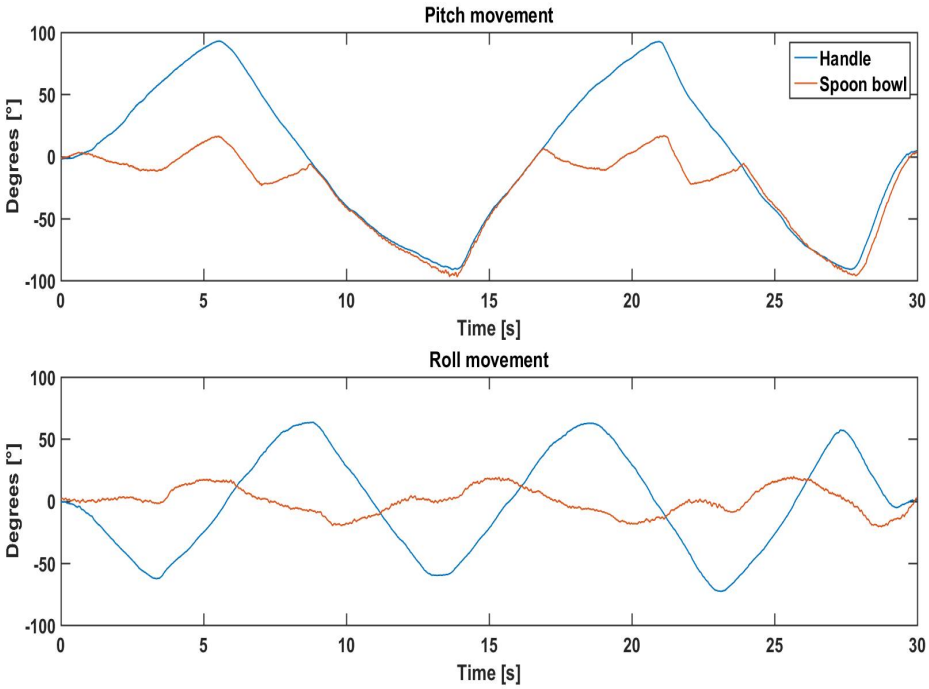


Figure 3.17. Performance of the spoon's pitch and roll movement in slow oscillation, created in Matlab

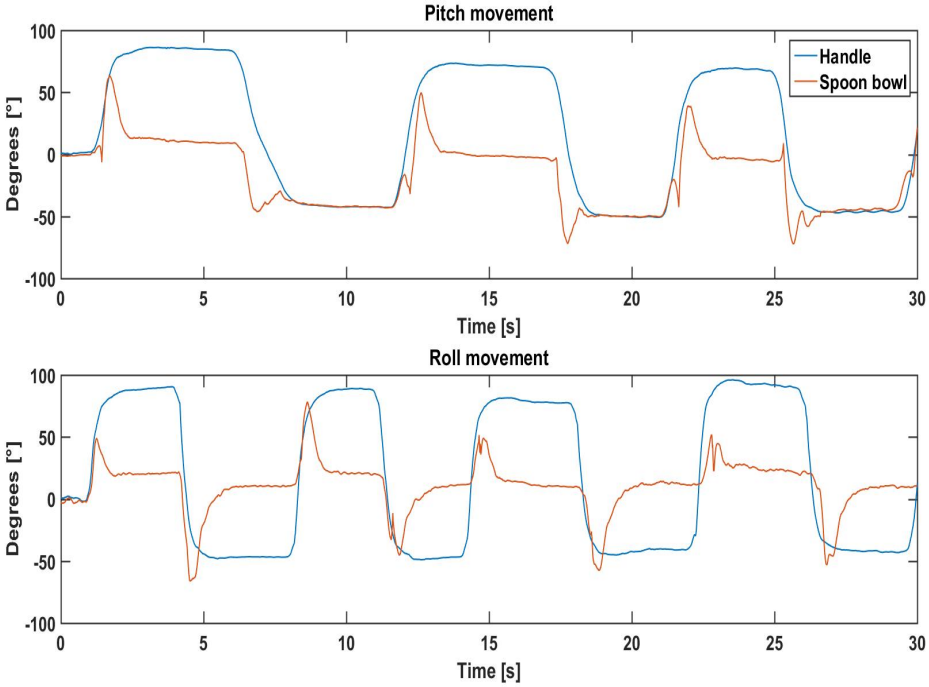


Figure 3.18. Performance of the spoon's pitch and roll movement in fast oscillation, created in Matlab

To evaluate and compare the performance of the two codes more clearly, the discrepancy ratio from the horizontal position (0°) were calculated for the worst case scenario in each graph. This can be visualized in Table 3.1 where 0% implies that the spoon bowl's position is strictly horizontal and 100% implies that the spoon bowl is rotating exactly as much as the handle.

Table 3.1. Overview of discrepancy

Code	Oscillation	Pitch discrepancy [%]	Roll discrepancy [%]
Similar project	slow	25	32
Own developed	slow	17	9
Similar project	fast	38	62
Own developed	fast	74	61

Chapter 4

Conclusions and discussion

The last constructed prototype followed the desired movements as it was intended with the thesis. However, the results and performance of the device are not satisfying enough to meet the project's initial demands when looking at the discrepancy in Table 3.1. Concerning high frequency tremors, the device can be improved a lot and a possible solution for this problem would be to use faster servo motors.

In Figure 3.13 and 3.17, where slow motions were applied, the spoon was struggling to hold a horizontal position, but although the spoon was not strictly holding a horizontal position, the deviation was not highly critical. With that being said, one can discuss the usability of the device for people with impaired motor skills. Even though the device would not be helpful for people who suffer from high frequency tremors, it might be helpful for people with loss of physical motor function. These people could be physically disabled or perhaps elderly people who have reduced motor skills.

To answer the research questions

> "How can an Arduino microcontroller be utilized to help people with impaired motor skills during their eating process?"
>
> - "How fast can the device react for a motion and will it be fast enough to help people with high frequency tremors?"
>
> - "To what extent will the device fit the hand of the user?"

the two sub-questions will first be taken into consideration.

The spoon does not react fast enough to be useful for people with high frequency tremors. The reason to this is that the motors are too slow to react for these motions. As the motors have a definite speed of rotation that is less than required

for high frequency tremors (or shaking in general), faster motors must be used to satisfy this requirement.

The device was constructed with a cylindrical handle with a diameter of 40 mm. This would likely fit most adult peoples hands. The device weighs around 130 grams in total, which is a reasonable weight considering what the spoon is intended to be used for.

After having the two sub-questions discussed, the main research question can be taken into consideration. The existing setup of the prototype with its limitation of not being able to handle high frequency tremors, might still be of use for people with impaired motor skills during their eating process. By looking at the discrepancy in Table 3.1, slow motions do get counteracted by the spoon with an acceptable deviation, and if food is placed on the spoon it would likely remain in the spoon bowl during these motions. This concludes that The Stabilizing Spoon can be utilized by people with impaired motor skills, though to a certain extent.

Chapter 5

Future work

The device, as stated in the conclusion and discussion-chapter, does not react fast enough to be of use for individuals with high frequency tremors. To meet this requirement other motors must be used, for example faster servomotors, DC-motors or perhaps stepper-motors would likely solve the problem. One alternative would be to use MG90S servo motors. The MG90S has roughly the same dimensions and weight as the SG90 but has a higher operating torque and higher speed rotation. When it is supplied with 6 V it is able to rotate 60 degrees in 0.08 seconds and have an output torque of 2.2 kgcm [engineering.tamu.edu, 2017]. If comparing the SG90 with its speed rotation of 0.1 s/60° and torque of 1.8 kgcm [micropik.com, 2014], the spoon with the MG90S servo motors would likely have good premises to compensate for high frequency tremors.

The size of the device is reasonable but quite large compared to ordinary utensils. A smaller device would therefore make a positive impact on the user in terms of comfort and discreteness. To do this the casing could be minimized in some areas due to hollowness in these places. The motors are unnecessarily large and an integrated rechargeable battery would minimize the required area for the 9 V battery which is now quite voluminous. The choice of an integrated rechargeable battery instead of the ordinary 9 V battery would also terminate the problem of changing batteries, which is quite impractical with the existing prototype. Rechargeable batteries would also be a benefit in environmental aspects, considering the waste of batteries when exchanging them.

Bibliography

[arduino.cc, 2016a] arduino.cc (2016a). arduino.cc. Available from: https://www.arduino.cc/en/Tutorial/PWM [cited 2017-04-14].

[arduino.cc, 2016b] arduino.cc (2016b). arduino.cc. Available from: https://www.arduino.cc/en/Tutorial/PWM [cited 2017-04-14].

[arduino.cc, 2017a] arduino.cc (2017a). Arduino nano. Accessed: 2017-04-14. Available from: https://www.arduino.cc/en/Main/arduinoBoardNano.

[arduino.cc, 2017b] arduino.cc (2017b). Arduino uno. Accessed: 2017-04-14. Available from: https://www.arduino.cc/en/main/arduinoBoardUno.

[Birtwell, 2016] Birtwell, K. (2016). Third year project report. Accessed: 2017-04-14. Available from: https://www.slideshare.net/KyleBirtwell/third-year-project-report-64623749.

[camfere.com, 2016] camfere.com (2016). camfere.com. Available from: http://www.camfere.com/stabilizers-gimbals-2320/p-d4027.html?currency=SEK&aid=cfgplase&gclid=CjwKEAjw_bHHBRD4qbKukMiVgU0SJADr08ZZi5HUodNdJNfCTSR83EHwUqN7Kc7hz0TVf4uuOYTYyxoC-Fvw_wcB [cited 2017-04-14].

[dx.com, 2017] dx.com (2017). Mpu6050. Accessed: 2017-04-14. Available from: http://img.dxcdn.com/productimages/sku_154602_2.jpg.

[electronics lab.com, 2017] electronics lab.com (2017). Arduino nano. Accessed: 2017-04-14. Available from: http://www.electronics-lab.com/make-your-own-arduino-nano-diy-arduino-nano/.

[engineering.tamu.edu, 2017] engineering.tamu.edu (2017). Mg90s servo, metal gear with one bearing. Accessed: 2017-06-03. Available from: https://engineering.tamu.edu/media/4247823/ds-servo-mg90s.pdf.

[Higgins, 1975] Higgins, W. T. (1975). A comparison of complementary and kalman filtering. Available from: https://www.google.se/search?q=complementary+filter&oq=comple&aqs=chrome.0.69i59j69i57j0l2j69i61j69i60.2229j0j7&sourceid=chrome&ie=UTF-8 [cited 2017-04-14].

[invensense.com, 2017] invensense.com (2017). Mpu6050. Accessed: 2017-04-14. Available from: https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

[Khazane et al., 2015] Khazane, N., Tscheope, A., and Watts-Willis, I. (2015). End of project documentation, a team 4 report. Accessed: 2017-04-14. Available from: http://www.csus.edu/indiv/t/tatror/senior_design/sd%20f14-s15%20reports/team_4_steadyspoon_f14_to_s15.pdf.

[learn.adafruit.com, 2017] learn.adafruit.com (2017). opengl-tutorial.org. Accessed: 2017-05-14. Available from: https://learn.adafruit.com/analog-feedback-servos/about-servos-and-feedback.

[liftware.com, 2016] liftware.com (2016). liftware.com. Available from: https://www.liftware.com/steady/ [cited 2017-04-14].

[madsci.org, 1998] madsci.org (1998). Gimbal lock. Accessed: 2017-05-15. Available from: http://www.madsci.org/posts/archives/aug98/896993617.Eg.r.html.

[MantechElectronics, 2017] MantechElectronics (2017). Micro servo 9g fs90. Accessed: 2017-05-16. Available from: http://www.micropik.com/PDF/SG90Servo.pdf.

[MathsPoetry, 2009] MathsPoetry (2009). Gimbal lock. Available from: https://commons.wikimedia.org/wiki/File:Gimbal_lock.png [cited 2017-04-14].

[mathworks.com, 2017] mathworks.com (2017). mathworks. Accessed: 2017-05-19. Available from: mathworks.com.

[micropik.com, 2014] micropik.com (2014). micropik.com. Accessed: 2017-05-16. Available from: http://www.micropik.com/PDF/SG90Servo.pdf.

[opengl tutorial.org, 2017] opengl tutorial.org (2017). opengl-tutorial.org. Accessed: 2017-05-5. Available from: .opengl-tutorial.org/intermediate-tutorials/tutorial-17-quaternions/.

[opensource.com, 2017] opensource.com (2017). What is an arduino? Accessed: 2017-04-14. Available from: https://opensource.com/resources/what-arduino.

[Orsini, 2014] Orsini, L. (2014). Arduino vs. raspberry pi: Which is the right diy platform for you? Accessed: 2017-05-16. Available from: http://readwrite.com/2014/05/07/arduino-vs-raspberry-pi-projects-diy-platform/.

[Rowberg and Poynter, 2016] Rowberg, J. and Poynter, W. (2016). Gyro stabilizer with arduino and servo. Available from: http://www.instructables.com/id/Gyro-Stabilizer-W-Arduino-and-Servo/ [cited 2017-06-03].

[Van de Maele, 2013] Van de Maele, P.-J. (2013). Reading an imu without kalman: The complementary filter. Accessed: 2017-04-14. Available from: `http://www.pieter-jan.com/node/11`.

Appendix A

Similar project code

The code used from the similar project is attached below.

```
//============ THIS CODE HAS BEEN MODIFIED============================
//This code was used in the bachelor's thesis, "The Stabilizing Spoon".
//The code was modified in the later section to enable one of the
//servo motors to rotate further than what was possible with the
//original code.
//
//Author: Johan Danmo and Johan Abrahamsson
//Name of the project: The Stabilizing Spoon
//Name of the program: CMAST
//TRITA NUMBER: MMK 2017:21 MDAB 639
//Date: 2017-05-19
//====================================================================
//
//2 servo planar stabilization system
//wp
//Jan 2016
//
//Based on Jeff Rowber's work found at
//https://github.com/jrowberg/i2cdevlib/blob/master/Arduino/MPU6050/MPU6050.cpp
//
//Use at your own risk.
//
//This code is placed under the MIT License (MIT)
//
//Copyright (c) 2016 woojay poynter
//http://www.instructables.com/id/Gyro-Stabilizer-W-Arduino-and-Servo/

//Permission is hereby granted, free of charge, to any person obtaining
//a copyof 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.

// Servo Connection
// BROWN - gnd
// red - 5v
// yellow - d10 (pwm on Sero 1)
//        - d11 (servo 2)

// MPU Connection
//
// VCC - 5v
// GND - GND
// SCL - A5 (w/ 10k PuR)
// SDA - A4 (w/ 10k PuR)
// INT - D2 (not used)

#include <Servo.h>
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#include "Wire.h"


#define LED_PIN 13
bool blinkState = true;

Servo Servo1; // First Servo off the chassis
Servo Servo2; // Second Servo off the chassis

int Servo1Pos = 0;
int Servo2Pos = 0;

float mpuPitch = 0;
float mpuRoll = 0;
float mpuYaw = 0;

const int MPU_addr=0x68; // I2C address of the MPU-6050
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
float BOX;
```

```
// define MPU instance
MPU6050 mpu;                 // class default I2C address is 0x68;
    specific I2C addresses may be passed as a parameter here

// MPU control/status vars
uint8_t mpuIntStatus; // holds actual interrupt status byte from MPU
uint8_t devStatus;    // return status after each device operation (0 =
    success, !0 = error)
uint16_t packetSize;  // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;   // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q;         // [w, x, y, z]       quaternion container
VectorInt16 aa;       // [x, y, z]          accel sensor measurements
VectorInt16 aaReal;   // [x, y, z]          gravity-free accel sensor
    measurements
VectorInt16 aaWorld;  // [x, y, z]          world-frame accel sensor
    measurements
VectorFloat gravity;  // [x, y, z]          gravity vector
float ypr[3];         // [yaw, pitch, roll] yaw/pitch/roll container and
    gravity vector

// relative ypr[x] usage based on sensor orientation when mounted, e.g.
    ypr[PITCH]
#define PITCH  1     // defines the position within ypr[x] variable for
    PITCH; may vary due to sensor orientation when mounted
#define ROLL 2     // defines the position within ypr[x] variable for ROLL;
    may vary due to sensor orientation when mounted
#define YAW  0     // defines the position within ypr[x] variable for YAW;
    may vary due to sensor orientation when mounted

// ===============================================================
// ===                      INITIAL SETUP                      ===
// ===============================================================

void setup()
{

  Servo1.attach(10); // attaches the servo on D11 to the servo object
  Servo2.attach(11); // Second servo on D11

  Wire.begin();
    #if ARDUINO >= 157
  Wire.setClock(400000UL); // Set I2C frequency to 400kHz
  #else
    TWBR = ((F_CPU / 400000UL) - 16) / 2; // Set I2C frequency to 400kHz
```

```
#endif
Wire.beginTransmission(MPU_addr);
Wire.write(0x6B); // PWR_MGMT_1 register
Wire.write(0);    // set to zero (wakes up the MPU-6050)
Wire.endTransmission(true);
Serial.begin(115200);
delay(100);

//setup starting angle
Wire.beginTransmission(MPU_addr);
Wire.write(0x3B); //3 starting with register 0x3B (ACCEL_XOUT_H)
Wire.endTransmission(false);
Wire.requestFrom(MPU_addr,14,true); // request a total of 14 registers
AcX=Wire.read()<<8|Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C
    (ACCEL_XOUT_L)
AcY=Wire.read()<<8|Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E
    (ACCEL_YOUT_L)
AcZ=Wire.read()<<8|Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40
    (ACCEL_ZOUT_L)
Tmp=Wire.read()<<8|Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
GyX=Wire.read()<<8|Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44
    (GYRO_XOUT_L)
GyY=Wire.read()<<8|Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46
    (GYRO_YOUT_L)
GyZ=Wire.read()<<8|Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48
    (GYRO_ZOUT_L)


while (!Serial);    // wait for Leonardo enumeration, others continue
    immediately

// initialize device
Serial.println(F("Initializing I2C devices..."));
mpu.initialize();

// verify connection
Serial.println(F("Testing device connections..."));
Serial.println(mpu.testConnection() ? F("MPU6050 connection successful")
    : F("MPU6050 connection failed"));

// load and configure the DMP
Serial.println(F("Initializing DMP"));
devStatus = mpu.dmpInitialize();


// INPUT CALIBRATED OFFSETS HERE; SPECIFIC FOR EACH UNIT AND EACH
    MOUNTING CONFIGURATION!!!!
mpu.setXGyroOffset(118);
mpu.setYGyroOffset(-44); //-44
```

```
  mpu.setZGyroOffset(337);
  mpu.setXAccelOffset(-651);
  mpu.setYAccelOffset(670); //670
  mpu.setZAccelOffset(1895);

  // make sure it worked (returns 0 if so)
  if (devStatus == 0)
  {
    // turn on the DMP, now that it's ready
    Serial.println(F("Enabling DMP"));
    mpu.setDMPEnabled(true);

    // enable Arduino interrupt detection
    Serial.println(F("Enabling interrupt detection (Arduino external
        interrupt 0)"));
    mpuIntStatus = mpu.getIntStatus();

    // get expected DMP packet size for later comparison
    packetSize = mpu.dmpGetFIFOPacketSize();
  }
  else
  {
    // ERROR!
    // 1 = initial memory load failed, 2 = DMP configuration updates
        failed (if it's going to break, usually the code will be 1)
    Serial.print(F("DMP Initialization failed code = "));
    Serial.println(devStatus);
  }

  // configure LED for output
  pinMode(LED_PIN, OUTPUT);
} // setup()



// ================================================================
// ===                    MAIN PROGRAM LOOP                     ===
// ================================================================

void loop(void)
{

  // Get INT_STATUS byte
  mpuIntStatus = mpu.getIntStatus();

  // get current FIFO count
  fifoCount = mpu.getFIFOCount();

  // check for overflow (this should never happen unless our code is too
```

```
    inefficient)
if ((mpuIntStatus & 0x10) || fifoCount == 1024)
{
  // reset so we can continue cleanly
  mpu.resetFIFO();
  Serial.println(F("FIFO overflow!"));
  return;
}

if (mpuIntStatus & 0x02) // otherwise continue processing
{
  // check for correct available data length
  if (fifoCount < packetSize)
    return; // fifoCount = mpu.getFIFOCount();

  // read a packet from FIFO
  mpu.getFIFOBytes(fifoBuffer, packetSize);

  // track FIFO count here in case there is > 1 packet available
  fifoCount -= packetSize;

  // flush buffer to prevent overflow
  mpu.resetFIFO();

  // display Euler angles in degrees
  mpu.dmpGetQuaternion(&q, fifoBuffer);
  mpu.dmpGetGravity(&gravity, &q);
  mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
  mpuRoll = ypr[PITCH] * 180 / M_PI;
  mpuYaw = ypr[YAW] * 180 / M_PI;

  // flush buffer to prevent overflow
  mpu.resetFIFO();

  // blink LED to indicate activity
  blinkState = !blinkState;
  digitalWrite(LED_PIN, blinkState);

  // flush buffer to prevent overflow
  mpu.resetFIFO();

Wire.beginTransmission(MPU_addr);
Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
Wire.endTransmission(false);
Wire.requestFrom(MPU_addr,14,true); // request a total of 14 registers
AcX=Wire.read()<<8|Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C
    (ACCEL_XOUT_L)
AcY=Wire.read()<<8|Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E
    (ACCEL_YOUT_L)
```

```
 AcZ=Wire.read()<<8|Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40
     (ACCEL_ZOUT_L)
 GyX=Wire.read()<<8|Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44
     (GYRO_XOUT_L)
 GyY=Wire.read()<<8|Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46
     (GYRO_YOUT_L)
 GyZ=Wire.read()<<8|Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48
     (GYRO_ZOUT_L)

//Locking motion for the pitch when tilting the spoon rear-downward.
 if ((ypr[ROLL] * 180 / M_PI)+90 > 120 and (ypr[ROLL] * 180 / M_PI)+90 <
     180)
 {
 mpuPitch = ypr[ROLL] * 180 / M_PI;
 if (AcY > 28152)
 {
   mpuPitch = 173 - 90;
 }
 }

 //Incrementation of degrees for the pitch motion.
 mpuPitch = floor(mpuPitch);
 if (135 <= (mpuPitch + 90) && (mpuPitch + 90) < 140)
 {
   BOX = 2;
 }
 if (135 <= (mpuPitch + 90) && (mpuPitch + 90) < 140)
   {
     BOX = 4;
   }
   if (140 <= (mpuPitch + 90) && (mpuPitch + 90) < 145)
   {
     BOX = 6;
   }
   if (145 <= (mpuPitch + 90) && (mpuPitch + 90) < 148)
   {
     BOX = 7;
   }
   if (148 <= (mpuPitch + 90) && (mpuPitch + 90) < 150)
   {
     BOX = 8;
   }
   else if (150 <= (mpuPitch + 90) && (mpuPitch + 90) < 155)
   {
     BOX = 11;
   }
   else if (155 <= (mpuPitch + 90) && (mpuPitch + 90) < 158)
     {
       BOX =13;
```

```
    }
  else if (158 <= (mpuPitch + 90) && (mpuPitch + 90) < 160)
    {
      BOX =15;
    }
  else if (160 <= (mpuPitch + 90) && (mpuPitch + 90) < 165)
    {
      BOX =17;
    }
  else if (165 <= (mpuPitch + 90) && (mpuPitch + 90) < 168)
    {
      BOX =19;
    }
  else if (168 <= (mpuPitch + 90) && (mpuPitch + 90) < 170)
    {
      BOX =21;
    }

  else if (170 <= (mpuPitch + 90))
    {
      BOX =23;
    }

  //Command the servo motors
  Servo2.write(mpuPitch + BOX-40);
  Servo1.write(-mpuRoll + 100);


  // flush buffer to prevent overflow
  mpu.resetFIFO();

 }
}
```

Appendix B

The project's own developed code

The code developed for this thesis is attached below.

```
//This code was used in the bachelor's thesis, "The Stabilizing Spoon".
//The code was modified in the later section to enable one of the
//servo motors to rotate further than what was possible with the
//original code.
//
//Author: Johan Danmo and Johan Abrahamsson
//Name of the project: The Stabilizing Spoon
//Name of the program: CMAST
//TRITA NUMBER: MMK 2017:21 MDAB 639
//Date: 2017-05-19

// ============= INCLUDING LIBRARIES AND DECLARING VARIABLES =============
#include<Wire.h>
#include <Servo.h>
const int MPU_addr=0x68; // I2C address of the MPU-6050
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;

float delta_t = 0.005;
float pitchAcc,rollAcc, pitch, roll, pitched;
float P_CompCoeff= 0.98;

// ============= INITIAL SETUP ==========================================
Servo myservo1, myservo2;
void setup(){
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B); // PWR_MGMT_1 register
  Wire.write(0);    // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
  Serial.begin(115200);

  myservo1.attach(10);
  myservo2.attach(11);
```

```
}

// ============= MAIN LOOP ================================================
void loop(){
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr,14,true); // request a total of 14 registers
  AcX=Wire.read()<<8|Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C
      (ACCEL_XOUT_L)
  AcY=Wire.read()<<8|Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E
      (ACCEL_YOUT_L)
  AcZ=Wire.read()<<8|Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40
      (ACCEL_ZOUT_L)
  GyX=Wire.read()<<8|Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44
      (GYRO_XOUT_L)
  GyY=Wire.read()<<8|Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46
      (GYRO_YOUT_L)
  GyZ=Wire.read()<<8|Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48
      (GYRO_ZOUT_L)

  //Complementary filter
  long squaresum_P=((long)GyY*GyY+(long)AcY*AcY);
  long squaresum_R=((long)GyX*GyX+(long)AcX*AcX);
  pitch+=((-AcY/40.8f)*(delta_t));
  roll+=((-AcX/45.8f)*(delta_t)); //32.8
  pitchAcc= atan((AcY/sqrt(squaresum_P))*RAD_TO_DEG);
  rollAcc =atan((AcX/sqrt(squaresum_R))*RAD_TO_DEG);
  pitch =(P_CompCoeff*pitch + (1.0f-P_CompCoeff)*pitchAcc);//pitch
      =P_CompCoeff*pitch + (1.0f-P_CompCoeff)*pitchAcc;
  roll =(P_CompCoeff*roll + (1.0f-P_CompCoeff)*rollAcc);

  /*
  if-statements to make the roll command go to where it is meant to go,
  i.e clockwise/counterclockwise rotation
  */
  if (pitch < -158)
  {
    pitched = abs(pitch + 158);
    pitched = pitched - 158;
  }
  else if (pitch > -156)
  {
    pitched = abs(156 + pitch);
    pitched = -156 - pitched;
  }

  //locked movement for upward direction of pitch
  if (pitched < -240)
```

```
  {
    pitched = -240;
  }

  //Servo commands, roll/pitch + nr, where nr is compensation for mounting
      to start horizontally
  myservo1.write((roll + 120));
  myservo2.write(pitched + 340);
}
```

TRITA MMK 2017-21 MDAB 639