Project in Electrical Engineering

Electric Skateboard

Vincent Ricknell
Anthon Jonsson
Abstract

Project in Electrical Engineering; Electric Skateboard

Anthon Jonsson & Vincent Ricknell

The purpose of this project was to improve the comprehension of theoretical and practical aspects of electronics. The idea was to create a practical product with real world use, with the use of programming, designing electronics and implementing it with mechanical parts, in a mechatronic project.

This report describes one solution to implement an electrical drive system to a regular skateboard. This includes both the theoretical and practical spectres on what to encounter and the reflections of the creation. Everything from wireless connections, programming, mechanical issues to choosing the right motor and power electronics.

The result was a vehicle for fast and flexible transport for shorter distances with respectable speed and range. Even though there are some mechanical difficulties while building an electric skateboard, the potential for further development regarding the board with more functions and mechanical strength.
Project in Electrical Engineering

Electric Skateboard

Executed by:
Anthon Jonsson
Vincent Ricknell

Uppsala
2017-05-31
1 Abstract

The purpose of this project was to improve the comprehension of theoretical and practical aspects of electronics. The idea was to create a practical product with real world use, with the use of programming, designing electronics and implementing it with mechanical parts, in a mechatronic project.

This report describes one solution to implement an electrical drive system to a regular skateboard. This includes both the theoretical and practical spectres on what to encounter and the reflections of the creation. Everything from wireless connections, programming, mechanical issues to choosing the right motor and power electronics.

The result was a vehicle for fast and flexible transport for shorter distances with respectable speed and range. Even though there are some mechanical difficulties while building an electric skateboard, the potential for further development regarding the board with more functions and mechanical strength.
# Contents

1. Abstract  

2. Introduction  

3. Theory  
   3.1 Serial communication  
   3.1.1 I2C protocol  
   3.2 Pulse width modulation  
   3.3 Battery alternatives  
   3.3.1 LiPo battery specifications  
   3.4 Arduino  
   3.5 Bluetooth  
   3.5.1 HC-05 module  
   3.6 DC motors  
   3.6.1 BLDC motor  
   3.6.2 Power electronics  

4. Method and Materials  
   4.1 Remote control design  
   4.1.1 Remote controller code  
   4.2 Scaling and acceleration  
   4.3 Bluetooth/serial-communication and coding  
   4.4 Motor selection  
   4.4.1 Motor controller selection  
   4.5 Battery selection  
   4.6 Driving-belt/gear and motor mount  
   4.7 Assembly  
   4.7.1 Mounting gear and motor  
   4.7.2 Power electronics/Arduino/battery  

5. Results  
   5.1 Driving system  
   5.2 The controller  
   5.3 Speed  
   5.4 Bluetooth (delay)  
   5.5 Scaling  

6. Discussion  
   6.1 Error management and safety  
   6.2 Bluetooth and code  

7. Conclusion  

A Arduino code
2 Introduction

Skateboards have been a popular mean of transportation among the younger generation. The simplicity of the device has always been the key to its success. By pushing with your legs you create the mechanical force needed to gain speed.

As the technology has advanced a lot since the 1950s, it was appropriate step in the right direction to implement a modern take of the skateboard. Making an electronic skateboard controlled without wires, is a step in that direction. The purpose of an electrical skateboard is to enable a fast way to transport a single person the last bit of a commute as a complementary device to other public transportation.
3 Theory

3.1 Serial communication

Serial communication is a method to send and receive data (more than 1 bit) on a single line, all this through a transmitting port (TX) and a receiving port (RX). Compared to parallel communication, where 8 data bits would require 8 cables with 8 canals on both sending and receiving end (illustrated in figure 1).

Serial communication is the most common method to send and receive data. Two of the more well known serial interfaces are USB and Ethernet. Among the group of serial communication SPI and I2C are common interfaces. Serial interfaces can be divided into two groups; Synchronous and Asynchronous serial communication.

Synchronous is the type of serial communication where all devices are connected together through their data lines with a collective system clock. This often makes the system itself more stable and reliable. However, one extra wire between all the devices to connect them are necessary. Asynchronous on the other hand, operates without any needs for a clock signal.

Baud rate describes how many bits are sent over the serial wire per second (bps). The baud rate is specified upon how fast the system needs to be. The limits are not strict. A typical micro controller as an Arduino is not able to operate with faster rates than 115200 bps.
3.1.1 I2C protocol

I2C, also known as Inter-Integrated Circuit, is a protocol which allows more than one slave to be attached to the same two lines connected to a master.

![I2C Protocol Diagram](image)

Figure 2: I2C protocol illustrated. [9]

The wires illustrated in figure 2 are $V_{dd}$ which is the voltage fed to each and every device, SDA (Serial Data Line) where the data is transferred and SCL (Serial Clock), making I2C an example of a synchronous serial communication protocol. The I2C protocol works in that manner that the master writes a start bit and a 7-bit address, where the address is the specific slave the master wants to communicate with. The 8 bit stripe is followed by either a 0 or 1 whether the master wants to write or read. The slave (if the address exist) response with sending an ACK bit (acknowledgement). The master ends the transmission with either write STOP to end the communication completely or write START to alter control over the I2C bus for another transmission.

3.2 Pulse width modulation

Pulse width modulation, also known as PWM, is a very common modulation technique in all sorts of electronics.

PWM is a way to vary the power fed to a device or a given load. The power is controlled by switching the voltage on and off and creating a relation between the off-time and the on-time (and in relation to the period time). The higher time it’s switched on compared to off, the higher power is fed to the load. To control the power sent to the load, the ‘duty cycle’ is being changed. The duty cycle describes the relation between the on and off time. Higher duty cycle equals higher power. A low duty cycle (<50%) equals longer off-time than on-time, and vice versa. The duty cycle is often declared in percent, where 0% is completely off and 100% is completely on. An example of this is demonstrated in figure 3.

![PWM Modulation Diagram](image)

Figure 3: PWM modulation with 25% and 50% duty cycle.
3.3 Battery alternatives

In recent years the alternatives for power sources for wireless devices as for electric vehicles has evolved in both energy density and especially the costs per Watt-hour. The alternatives for consumer-grade rechargeable battery chemistry’s with high energy density includes NiMh (Nickel-metal hydride), LiPo (Lithium polymer), LiFePo4 (Lithium iron phosphate) and LiCoO2 (Lithium cobalt oxide). Lithium cobalt oxide is the most common type of Lithium ion cell.

<table>
<thead>
<tr>
<th>Chemistry</th>
<th>NiMh</th>
<th>LiPo</th>
<th>LiFePo4</th>
<th>LiCoO2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal Voltage (V)</td>
<td>1.2</td>
<td>3.7</td>
<td>3.2</td>
<td>3.6</td>
</tr>
<tr>
<td>Energy density (Wh/kg)</td>
<td>60-120</td>
<td>&gt;140</td>
<td>90-120</td>
<td>150-250</td>
</tr>
<tr>
<td>Peak dischargerating (c)</td>
<td>5</td>
<td>&gt;35</td>
<td>&gt;30</td>
<td>2</td>
</tr>
<tr>
<td>Cycle life (80% DoD(^1))</td>
<td>300-500</td>
<td>500-1000</td>
<td>1000-2000</td>
<td>500-1000</td>
</tr>
<tr>
<td>Charge time (h)</td>
<td>2-4</td>
<td>1-2</td>
<td>1-2</td>
<td>2-4</td>
</tr>
<tr>
<td>Internal resistance</td>
<td>Low</td>
<td>Very low</td>
<td>Very low</td>
<td>Moderate</td>
</tr>
<tr>
<td>Self discharge/month (at room temp)</td>
<td>30%</td>
<td>&lt;5%</td>
<td>&lt;5%</td>
<td>&lt;5%</td>
</tr>
</tbody>
</table>

Table 1: Table of battery performance. [1]

The discharge rating of a battery is directly proportional to the capacity of the cell. With a given capacity and C constant, the delivered current from a single cell can be calculated with equation (1).

\[ A = C \times Ah \]  

3.3.1 LiPo battery specifications

The lithium polymer cell shares similarities to the Lithium ion battery cell which is made of cobalt oxide cathode, graphite carbon as the anode and lithium salt as the electrolyte. The lithium polymer uses a solid polymer electrolyte that provides the battery with high discharging abilities but also lowers it energy density compared to the lithium ion cell. Lithium Polymer battery cells operating voltage range from 3.3V to 4.2V, with nearly a linear discharge curve between voltages 4.2V to 3.5V at a discharge of 2C or lower. As many other types of chemistry’s the Lithium Polymer cell life cycle span benefits if the dept of discharge is low and always below 80%. To charge the battery cell a constant current source is needed with a maximum voltage of 4.2V. The charge current is normally at 1C but specialized cells can accept higher charge currents. The drawback with higher charge current is a vastly shorter cell cycle life.

\(^1\)Dept of discharge, percentage of the depletion of the total capacity of the battery cell. The battery’s cycle life is defined as the amount of 80% discharge cycles until the battery's remaining capacity is 70% of the original amount.
3.4 Arduino

Arduino is a variation of open source single-board microprocessors and micro controllers with a broad field of application in small electronic projects or even industrial implementations. The Arduino boards use Atmega type micro controllers which has a various analog and digital ports used as input and output (I/O) as well as PWM-generating outputs. The Arduino board provides easily reachable connections as well as in some boards voltage regulators and protection circuits. Some of the ports support serial communication, and the majority of Arduino boards features USB to serial conversion allowing it to connected to a computer via an USB connection. The 8-bit Atmel AVR chip supports flash memory storage which the user uploaded integrated development environment (IDE) is stored. The IDE is a C/C++ based program which has many different ready made libraries which makes the code easily reachable to novice users.

3.5 Bluetooth

The Bluetooth technology is a well established radio standard originally invented by Swedish company Ericsson Mobile in late 1980s and is today a very well entrenched technology used in many smart devices. Bluetooth uses high frequency radio waves as a carrier frequency between 2.4 to 2.485 GHz. The Bluetooth connection is established between a master device and at least one slave device. The data is transmitted via small packages during a connection event. The data is interpolated with the carrier frequency which produces a modulated signal where the binary zeroes deviates a negative frequency change while the binary ones gives a positive deviation. Bluetooth uses frequency-hopping technology, Frequency-hopping spread spectrum (FHSS) to minimize interference caused by other transmitting units that also use the same frequency range. This works by switching between 79 predetermined frequency channels decided by the master device, which occurs in a standardized rate of 1600 changes per second but can be changed to fit specified energy consumption.

During the connection event the master and slave device alternates between sending and receiving data with at least one data package sent from the master device and one acknowledge data package sent from the slave in return. When the connection event is over the devices changes frequency to the next predetermined channel. If one data package is lost during transmission due to a stronger interfering frequency the master will not receive the acknowledge package from the slave. This results in the master to resend the lost package in the next connection event when the frequency has changed and the interference is likely not present. This switching of channels provides the system an excellent protection from data loss in the event of interference, and similar systems are also seen in various serial protocols like I2C. The Bluetooth standard shares it’s frequency spectrum with a few commonly available systems such as Wi-Fi or remote controlled toys since the 2.4 GHz band is a unlicensed frequency band. The high resistance to interference is therefore of high importance to reliably use Bluetooth for data transmission.

Bluetooth can be implemented with common micro controllers such as Arduino to introduce wireless functionality, as between a Bluetooth enabled computer, smart phone or even other micro controllers.

3.5.1 HC-05 module

An example of a module that can communicate with Arduino based micro controllers via UART is the Bluetooth enabled module HC-05. HC-05 uses the Bluetooth protocol SPP (Serial Port Protocol) which enables it communicate to a micro controller with UART serial communication. The UART data rate, or baud rate, can be programmed to up to 460800 bits per second. The Bluetooth version is 2.0+EDR (Enhanced Data Rate) which enables it to reach transmission speed up to 3Mb per second. The HC-05 module can either be programmed as a master or slave unit via AT-commands. The module has a low power consumption and a relative compact size of 12.7x27mm which makes it suitable for small systems with limited size. The device has a built in 2,4 GHz antenna and supports 12 programmable IO ports.
3.6 DC motors

DC motors are a collection of electronic devices that transform direct electric energy flow to mechanical energy. Usually this includes using the form of magnetic field. Two common subcategories are BLDC and Brushed DC motor.

3.6.1 BLDC motor

BLDC (brushless DC-motor) is an high efficient electronic motor used in a broad field such as in computer hard drives, cooling fans, radio controlled toys, and CD/DVD players.

The theory behind the BLDC motor is relatively simple. As shown in figure 4 the stator is constructed of poles wound in pairs on opposing sides of each other and a permanent magnet as a rotor. There are variations of BLDC motors where the rotor is either rotating inside the motor (as the 4 shows) or rotating on the outside of the motor. There are variations of one phase and two phase BLDC motors but the efficiency is increased with each additional phase since it distributes the force more evenly, but more phases increases the complexity and cost of the motor. This results in 3 phase motors being the most cost effective alternative and thus the most commonly available in electric vehicles and high power applications. Controlling a BLDC motor requires a phase current controller which sends currents to each winding to induce a rotating magnetic field which creates a rotational force on the rotor causing it to rotate. BLDC motors can also be used as a generator where it produces a three phase sinusoidal wave. The stator is generally constructed with a iron core and the winding are as tightly wound as possible to increase the magnetic flux. The rotor is made with permanent magnets such as neodymium magnets which have a very strong static magnetic field. Typically the efficiency for a BLDC motor is between 85-90% but decreases with higher loads. Most of the losses is caused by a reduction in the magnetic field in the air gap between the rotor and stator, so a small gap is sought after in making efficient BLDC motors. The causes of the decreased efficiency with higher loads are the resistance in the winding causes a temperature rise with higher currents which lowers the strength of the permanent magnets in the rotor. One of the most significant ratings of a BLDC motor is the $K_v$ which is a constant that reflects the amount of back-emf which is induced at a specific motor speed. The $K_v$ is reciprocal to the back-emf constant $K_e$ accordingly:

$$K_v = \frac{1}{K_e} \quad (2)$$

An other important rating is the torque constant $K_t$ which provides information on how much torque the motor produces for a given current:

$$T = K_t \times I \quad (3)$$
3.6.2 Power electronics

Power electronics can be described as control and modulation/alternation of electric power. Some modern examples of these type of systems (simplified) are diodes, transistors, and thyristors.

A specific kind of power electric commonly used together with brushless DC motors (BLDC) is called ESC (short for electronic speed controller). An electronic speed controller creates a three-phase AC output, all this controlled by a PWM-signal sent to the input. To make sure that the right phase is inducted along with the motor rotation, one way to keep track of the rotation and position is to measure the amount of back-EMF, short for counter-electromotive force. This phenomenon is based on the magnetic field inducted by the current flowing through each phase. The voltage drop’s polarity is reversed compare to the input voltage, which creates a counter-magnetic field pushing against the current inducing it in the first place. This can be described with Faraday’s law of induction.

\[
\epsilon = -d\Phi_B / dt
\]

Where \( \epsilon \) is the EMF-voltage and \( \Phi_B \) is the inducted magnetic field. Other ways to keep track of the rotation is with either a hall sensor or an optical sensor. The hall effect, similar to the back-EMF, is based on the fact that a current going through a conductor initiates a magnetic field in which this magnetic field is perpendicular to the initiated current.

When controlling a BLDC motor with 3 phases, there are 60° shift between each phase. In order to adjust the speed of the motor, the frequency of the fed voltage needs to be adjusted, and the torque from the motor is linear to the current fed to it.

Figure 5: A simplification of power electronics, controlling a BLDC motor. [12]

Figure 5 shows a simplified version of a microprocessor with a 3-phase bridge controller. The microprocessor feeds signals to driver, where the driver sends PWM signals going to the 3-phase bridge. The bridge switches the PWM signals from the driver in a circular pattern (60° phase shifted from each other) to get the rotor to spin. In order to increase the speed of the motor, the frequency of the in going PWM signals are increased as well. The torque from the motor shaft is proportional to the current in each phase.
4 Method and Materials

4.1 Remote control design

The controller design was focused on comfort, ease of use, and providing the user with relevant information of the parameters of the system. The case of the controller was built with a modified RC-controller for cars which had a comfortable grip, enough space to contain the required electronics and already contained a high quality potentiometer connected to a throttle trigger. This ensured an easy way of controlling the speed by simply squeezing the trigger with the left index finger to achieve a desired power output. Inside the remote there are a battery to power it, Arduino micro, Bluetooth module for communication, potentiometer as user input, charging circuit for the battery, OLED-display, and a GRB LED-strip to be used as a graphical user interface. The easily-accessed charging circuit is to charge the battery with a commonly available micro USB cable, without the need to take the case apart. The LED-strip consists of four LED’s, which were programmed to work as an analog indication on how much power is provided to the motor from the controller. The LED:s indicates 4 stages where each stage symbolize 25% in the range 0-100%.

The OLED-display works as an more advanced UI and is connected as seen in figure 6 via i2c to the Arduino micro’s SDA and SCL ports. The screen displays the battery voltage of both the remote and skateboard in form in pillars which lengths changes dependent on the current battery percentage. Additionally, the input value of the trigger is displayed in it’s raw percentage value.

![Figure 6: Remote controller schematic.](image)

The battery in the remote was chosen as a one cell lithium polymer cell rated to 600mAh capacity. The small size and the ease to charge made it a good candidate to power the remote controller. Since the battery’s voltage range from 4.2V to 3.5V dependent on the remaining capacity. The Arduino micro, Bluetooth module and the OLED display are rated for 5V DC, meaning an DC-DC converter that outputs 5V was needed. The LiPo battery also had to be monitored to not get discharged to an damaging low voltage and required a charging unit to be charged while integrated into the remote.
An example of an DC-DC boost converter is the IC TSPS61200 which boosts voltages in the input interval 0.3V to 4.2V, and gives an selectable output voltage which include 5V with an high efficiency. The IC has also have an small quiescent current of approximately 55 $\mu$A and can easily be turned on and off by pulling an enable pin high or low. To charge the battery an IC such as the MCP73831 can be used as it delivers an selectable constant current, an selectable end-voltage and accuracy of 0.75%. To save time the pre made board "Power Cell" by SparkFun containing the mentioned IC’s. The board was implemented into the remote with one change, the enable pin of the TSPS61200 was connected to a pull-up resistor that was changed to a pull-down configuration in order to have the the switch pull the enable pin high (above $0.8 \times V_{in}$) and thus enabling the booster regulator.

4.1.1 Remote controller code

The Arduino code of the remote includes an read of the analog value of the potentiometer, which is remapped from the 10 bit ADC value (1024 integer values between 0-1023) to the 8 bit value (256 integer values between 0-255) that are sent via UART to the Bluetooth module. The values have to be remapped since Arduino’s digital communication can only process 8 bit values. Since the remote also displayed the voltage of the batteries inside the skateboard the value was read from the Bluetooth module and converted to a pillar indicating the percentage of charge. The voltage from the battery inside the remote is also read through an analog input and displayed similarly as the measured voltage of the skateboard batteries. The code used Adafruit libraries in order to control the WS2812B LED:s, a timer library was used in order for the battery voltage to be read less often to minimize the amount of noise that would otherwise be displayed.

4.2 Scaling and acceleration

Scaling the speed is necessary for comfort in this kind of electronic vehicles. By implementing an exponential velocity curve, the driving experience while driving the skateboard will end up a lot smoother and less twitchy.

![Exponential acceleration](image)

Figure 7: Exponential acceleration illustrated.

In practice, this typical exponential acceleration function won’t fully work. By examine figure 7 the vehicle won’t move for the first 4.5 seconds. What is needed is a combined ramp and exponential function. In other words, a function which is a bit steeper in the beginning (but still tranquil) and a bit flatter near the maximum speed.
4.3 Bluetooth/serial-communication and coding

The need of wireless communication and control as well as the need to eliminate or handle errors by coding, limited the options for this project. Bluetooth is the best option regarding a self-connected device who operates wirelessly and with serial communication.

There are some other options to choose from other than Bluetooth. RC-receiver and transmitter are an alternative, but the disadvantage is that there is no possibility to map or modify the potentiometer-values or voltage values through Arduino coding.

One of the drawbacks when working with Bluetooth and value mapping/modifying is delay. A responsive system with low delay controlling is necessary when building an electric skateboard purposely built for city cruising and fast reaction both when accelerating and braking. To measure the delay, a value was transmitted from one of the arduinos to the other, this transmission triggered a timer. When the second arduino received the value, it transmitted a specific value back to the first arduino. In tandem as the first arduino got the recalling value, the timer stopped and the delay was measured and noted.

4.4 Motor selection

Since the board was specified to reach a top speed of 25 km/h the specifications of the motor was calculated with the deciding factors that were chosen prior. These factors are the size of the wheel and the gearing ratio between the motor and wheel. The wheel had diameter of 65mm with the gearing ratio of 3:1. This meant that the wheel had to travel a distance of approximately 7 m each second at 25 km/h. With the diameter of 65mm the wheel had to rotate at 34.3 rotations per second, or 2056.7 rpm. With the gear ratio 3:1 the motor had to turn at 6170 rpm to achieve the required rotational speed at the wheel. The speed constant $K_v$ of the motor can roughly be used to calculate the motors rpm at a certain applied voltage with the equation (2) and with the voltage of 22.2V the $K_v$ constant of the motor had to be at least $\frac{6170}{22.2} = 278rmp/V$. With the known $K_v$ constant, the motor’s required power output had to be roughly calculated with some reasonable requirements. The motor had to have enough power to be able to drive approximately 80kg mass (weight of driver + skateboard) up a reasonable slope. With the assumption of a slope with the inclination of 20 ° the amount of force that is needed to overcome is equal to $\sin(\arctan(0,20)) \times 80kg \times 9.82m/s^2 = 154N$. The amount of power that is required to drive at 7 m/s with the force 154 N can be calculated with $P = F \times v$ which equals to $154N \times 7m/s = 1078W$. However this is neglecting the amount of force that caused by friction between the wheels and the asphalt but also the more substantial force that is caused by air resistance. For a very rough estimation of the amount of air resistance the drag equation $(F_D = \frac{1}{2} \rho v^2 C_d A)$ can be used with the information that the human body’s drag coefficient and density is roughly 1, and the average cross sectional area is $1.6m^2$ when standing on the side the force at $7m/s$ is approximately $40N$. Using this in the estimation increases the required power to $(154 + 40)N \times 7m/s = 1358W$. Because of other frictions such as in the driving belt and the wheels friction against the road the remaining losses was estimated to be around 10%, this means that the motor needed to have around 1500W of power to meet the defined requirements.

4.4.1 Motor controller selection

The motor controller was chosen together with the motor and battery and was matched to have enough margin to reliably controlling the motor. The minimum specifications was a continuous deliver of 60A and supporting 25.2V which is the voltage of the selected batteries. Since there was no sensor on the motor the controller had to measure the back emf produced from the motor in order to determine the switching frequency. The chosen controller was the X-Car Beast Series speed controller which supports up to 120A continuous current and short bursts of 720A. The controller also supports regenerative breaking which is important in effectively breaking the skateboard with higher loads as in high speed or when descending. The controller is easily controlled with a PWM signal with pulses between 1 to 2ms, where 1ms is a full stop and 2ms is 100% on. The controller also included a 6V regulator rated for 3A which was used to power the rest of the system in the board.
4.5 Battery selection

Since the chosen motor and controller had been chosen the choice of battery was primary what kind of chemistry which was best suited and what voltage was needed. The motor doesn’t draw a high amount of current in normal operation which means that almost all alternatives would work depending on their size. However in certain situations as when driving up hills or when quick acceleration is required currents up to 60A can be drawn from the motor. This narrows the selection as only a few types of batteries can deliver a high current with relatively low capacity. Since the battery was supposed to be mounted under the board the size was also taken into consideration, which meant that the energy density needed to be high in order for them to provide enough energy for the skateboard to be a viable transportation vehicle. The battery chemistry that met the specifications was the lithium polymer cell. The battery’s capacity was chosen to a size that could supply enough current without the cell’s voltage lowering to a damaging point. The amount of cells was specified by the maximum allowed voltage from the motor controller and was 25.2V which equals to 6 cells connected in series. The chosen battery cell’s capacity was 5Ah which means that the battery could theoretically deliver currents at 175A with the rated 35C in short bursts without taking damage.

4.6 Driving-belt/gear and motor mount

The driving belt and gear wheel connects the wheels with the BLDC motor with 3:1 gear reduction that increases the torque and decreases the speed on the one wheel drive system with a factor of 3. The toothed driving belt is made of a high grade plastic with a small amount of elasticity. The efficiency of a toothed driving belt is typically around 95% due friction in the elastic driving belt causing heat when in motion. The motor mount is constructed of aluminum and can be mounted to the trucks with 3 set screws. The motor is fastened to the plate with m3 screws and the length from the wheel can be adjusted in order to get the correct rigidity of the belt.
4.7 Assembly

Except for the arduino wiring and coding/programming, assembly was a big part in this project. Even though the part of software and hardware engineering is key to a safe and sustainable device, the physical parts needs to be proper built as well.

4.7.1 Mounting gear and motor

A robust construction of the gear and driving belt is mandatory for a safe contraption.

After the theoretical dimensions concerning the motor, driving belt, motor mount, gear wheels, and a motor was chosen. The chosen motor was the Turnigy Aerodrive SK3 BLDC motor with a maximum rated power of 1510 W. Though there are more aspects than the power to keep in mind while choosing a motor. The given specifications of the motor is the following:

<table>
<thead>
<tr>
<th>Specs</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>570 g</td>
</tr>
<tr>
<td>Power</td>
<td>1510 W</td>
</tr>
<tr>
<td>Kv</td>
<td>280 rpm/V</td>
</tr>
<tr>
<td>Length</td>
<td>33 mm</td>
</tr>
<tr>
<td>Diameter</td>
<td>49 mm</td>
</tr>
<tr>
<td>Resistance</td>
<td>33 mΩ</td>
</tr>
<tr>
<td>Max current</td>
<td>60 A</td>
</tr>
<tr>
<td>Max voltage</td>
<td>37 V</td>
</tr>
</tbody>
</table>

Table 2: Specifications of BLDC motor. [13]

The specs of the chosen motor (table 2) met the theoretical power that was previously calculated. The dimensions where well enough within of what would fit underneath the board. A small and compact motor with high power and high torque.

To attach the driving belt between the wheel and the motor, the gear wheels needed to be centered in the wheel axis and the rotation axis of the motor in order to prevent slack in the driving belt. The centering of the motor-axis was straight forward. The only thing that was needed was an adapter (6-8mm metal ring) between the motor shaft and the first gear wheel.

Figure 8: Driving belt and gear system.
The gear wheel was attached on the electric motor axis and fixed with help of 2 screws. To fix the other gear wheel to the skateboard wheel two holes needed to be drilled through the skateboard wheel. A lack of precision devices and to soft rubber in the skateboard wheel led to a poor centering of the gear wheel.

The 2nd step to fix the motor to the skateboard wheel through gearing was to attach the motor to the skateboard itself. In order to do this an aluminum plate was mounted to the skateboard truck. The plate was screwed on with three setscrews in to the metal of the skateboard truck in order to keep it tight and durable. Afterwards, the motor was fastened onto the metal plate and the driving belt was tightened by pulling the motor back as much as possible before it was mounted on with two screws.

4.7.2 Power electronics/Arduino/battery

The idea was to gather all electronics (incl battery, Arduino, battery and such) on the same place and easy to access. The mounting took place in a simple lunch box placed under the board, just in front of the driving system. The Arduino was connected to the Bluetooth module, speed controller and battery as shown in connection diagram shown in figure 9.

By screwing the box on to the board with four screws the box were tethered. The batteries were fixed in the box by bending two metal stripes in an ’U’ shape and fitting the batteries in place with a pair of rubber bands. The Arduino board was fixed with glue in the box to prevent any chances of short circuit.

Figure 9: Connection diagram of electronics in the skateboard.
5 Results

This is the section where all the results, regarding both theoretical and practical aspects are presented.

5.1 Driving system

The driving system is the weakest part of the construction. The gear wheel fixed on the skateboard wheel is off center, which leads to some slack in the driving belt. Because of the slack and the high torque motor, the acceleration needed to be scaled a bit more to prevent the belt from slipping and getting damaged. But because of the 3:1 gear wheels mounted on the skateboard wheel and the motor shaft, the board was able to start and accelerate in uphill.

5.2 The controller

The succeeded two-way communication led to an UI on the OLED-display with information about both the controller and the board. By removing a pull-up resistor and replacing it with a pull-down on the charging circuit, the voltage draw during off-mode was reduced and measured to be around 60 $\mu$A. Since the usable capacity in the remote is 80% of 600m$Ah$ the standby time is theoretically roughly 300 days. During use the Arduino, Bluetooth module and OLED display current draw was measured to around 100mA in use, so with the 600mAh battery the remote can be used for roughly 5 hours before it needs to be recharged. One important aspect of the remote usability is how easy it is to use and since the Bluetooth modules used was configured as a master and slave no further setup is required and the devices always connect with each other within seconds of turning the system on. Figure 10 illustrates an suggested use for the remote and demonstrates it OLED screen as well as the RGB LED’s on the top side.

Figure 10: Illustration of remote controller.
5.3 Speed

Since the beginning, the idea was to build a skateboard with a capacity of going with a speed of 28 km/h. With fully charged batteries and on a flat ground the highest speed that got registered was 26 km/h, which is within reasonable limit, close to the estimated speed.

5.4 Bluetooth (delay)

After testing the delay between the two arduino Bluetooth modules the delay itself were put out of consideration. When dealing with hand held controlling and wireless communication, a delay within the milliseconds spectra is very obtainable.

<table>
<thead>
<tr>
<th>Distance (m)</th>
<th>Delay (ms):</th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
<th>Average delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>38</td>
<td>46</td>
<td>41</td>
<td>41.7</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>38</td>
<td>40</td>
<td>40</td>
<td>39.3</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>37</td>
<td>44</td>
<td>43</td>
<td>41.33</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>38</td>
<td>39</td>
<td>43</td>
<td>40</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>44</td>
<td>44</td>
<td>41</td>
<td>43</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>42</td>
<td>43</td>
<td>40</td>
<td>41.7</td>
</tr>
</tbody>
</table>

Table 3: Table of Bluetooth delay.

By looking at table 3 the delay was completely out of cogitation. Both regarding the delay itself and the distance affecting the delay. Table 3 consist of measurements where the times are in a two way route, the actual delay between the remote and board can be given by halving the given measured values.

5.5 Scaling

As mentioned in subsection 4.2, it was necessary to scale the acceleration by implementing a function in the Arduino code, which scaled the value before feeding it to the ESC.

Figure 11: The coded "exponential" acceleration which was tested and plotted.

Figure 11 shows a plot over the PWM value sent to the ESC over time. This illustrates the exponential curve which is more ramped in the beginning and a bit flatter near the top speed.
6 Discussion

6.1 Error management and safety

In order to keep the electric board a safe vehicle it’s very important to keep error management in mind.

Because of the asynchronous 8 bit serial communication and lack of a collective system clock, the wireless Bluetooth communication is a source of errors. The asynchronous reading and writing causes the transmitting to be unreliable in some cases. In order to prevent these numbers to cause any fatal errors, most of the safety precautions is done in the Arduino coding. By help of the (already implemented) ramp function, spikes values in form of miss-read/writings are effectively filtered. Another critical situation when working with Bluetooth is the possibility of loss connection. Naturally, when a Bluetooth module shuts down or looses connection with the receiving end while transmitting data, the last value is sent. When the ESC receives the last sent value from the module, this value is sent in every loop to the motor, until a new value is sent to the ESC. With this in mind, the best thing to do is implementing a timer in the code, which keeps track on how many milliseconds there are between every sent value. When the system functions normally the measured time is very small, but in case of loss of transmission the loop does not update properly which causes the measured time to trigger an event that breaks the board to a full stop.

The main aspect when scaling the acceleration is the make the experience as smooth as possible, but there is (at least) one more reason to do this. The driving belt used in this project is made out of stronger plastic. The motor is, in this case, stronger and the risk is the torque will be too high. This is going to result in a slipping driving belt. In the worst case scenario the belt can break. By not having perfect centered gear wheels will also affect and put unwanted strain on the belt because of imbalance and the two wheels will rotate differently and out of sync.

6.2 Bluetooth and code

Wireless communication has been the main subject throughout this project. Bluetooth has enabled two-way communication, to create an UI on the controller with information about the battery level on the board.

Another alternative to this kind of project (electric skateboard) is to simply go with a RC transmitter and receiver. The RC serial communication operates a lot faster than the Bluetooth. A simple transmitter (controller) and receiver (ESC) removes any need of coding and micro controllers. But this also removes the possibility receiving any relevant information that is enabled with two-way communication. The option of having an UI with information from the board (wirelessly) is removed, due to the one-way communication from the transmitter to the receiver. In order to solve the problems with the speed ramping, there was an attempt at coding a smoothing filter in the Arduino code by continuously updating an array with a specific length with the last input value and then calculating the average value of the last values. However this was not possible with a single Arduino module that also communicated with the Bluetooth module since the calculations with the array introduced an substantial amount of delay into the code.
7 Conclusion

To build an electric skateboard is a matter of ideas and vision. There are so many different features and things that can be installed and implemented in these kind of projects. To begin with, a obvious improvement would be to center the gear wheels with more precise tools and get a stronger driving belt. This would improve the safety and enable the acceleration to be limited by the motor instead of the driving belt itself. To get an even stronger and more reliable driving system an additional motor could be added on the opposing wheel that would permit more acceleration and prolong the lifespan of the driving belt since it would reduce the amount of stress put on the belt.

Another thing to implement is cruise controller and PID regulator. The cruise control would be enabled by pushing a button on the controller to hold the current speed until it is released. By coding a PID regulator into the Arduino code that could read the current speed from a sensor it could have a feature of holding a constant speed even if the gradient or resistance would change.

Even though the mechanical complications that the implementation of an electrical motor to a skateboard created the overall performance of the board reached the expectations. Thanks to the boards relatively small size and light weight it is easy to bring on a bus or train which makes it an excellent means to transport a single person the last bit of a commute, and with its electrical drive system it can do so with high speed and with reliable performance.

Thanks to Johan Abrahamsson at the Devison of Electricity at Uppsala University for being our mentor during this project.
References

[1] Isidor Buchmann, *Comparison Table of Secondary Batteries.*
http://batteryuniversity.com/learn/article/secondary_batteries

http://batteryuniversity.com/learn/article/nickel_based_batteries

http://learningrc.com/motor-kv


https://www.bluetooth.com/specifications/bluetooth-core-specification/technical-considerations


https://en.wikipedia.org/wiki/Serial_communication

https://sv.wikipedia.org/wiki/I2C

http://www.mpoweruk.com/motorsbrushless.htm

http://www.learnengineering.org/2014/10/Brushless-DC-motor.html


A Arduino code

```c
#include <Adafruit_NeoPixel.h>
#include <SoftwareSerial.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "Timer.h"

#define LED 5
#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);

#define NUMFLAKES 10
#define XPOS 0
#define YPOS 1
#define DELTAY 2

const long analogPin1 = A1;
const long analogPin2 = A2;
const long analogPin3 = A3;

SoftwareSerial BTSerial(10, 11); // RX, TX
Timer t;
int long kontroll;
int long s;
float spanning;
float spn;
float just;
float lspn;
int rbatt;
int sbatt;
int x;
int n;
int longbatt;
String readString;
Adafruit_NeoPixel strip = Adafruit_NeoPixel(4, LED, NEO_RGB+NEO_KHZ800);

void setup()
{
  Serial.begin(9600);
  BTSerial.begin(115200);
  strip.begin();
  strip.show();
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setTextSize(1);
```

Figure 12: Arduino code for the controller (1/4).
```cpp
display.setTextColor(WHITE);
display.setCursor(25,4);
display.println("Electroboard");
display.setCursor(25,14);
display.println("Remote V.1.0");
display.display();
delay(2000);
display.clearDisplay();
battMeasure();
longbatt = 0;
}

void loop(){
    long gas = analogRead(analogPin);
    int regulator = analogRead(analogPin2);
    kontroll = map(gas, 0, 1023,0, 255);
    s = kontroll;
    BTSerial.write(kontroll);
    t.every(10000,battMeasure);
    t.every(10000,battBoard);
    t.every(50,updatePixel);

    if (BTSerial.available()){
        n = BTSerial.read();
        if (n<100 && n>0){
            Serial.println(n);
            longbatt = n;
        }
        else {
            Serial.println("corrup data");
        }
    }

    int d = map(s, 130 , 207 , 0, 100);
    if (s>128){
        if (d<10){
            display.clearDisplay();
            display.setTextSize(1);
            display.setTextColor(WHITE);
            display.setCursor(25,10);
            display.print("POWER:");
            display.print(d);
            display.println("%");
            int bar = rbatt/3.125;
```
int bar2 = sbatt/3.125;
display.fillRect(115, 32-bar, 8, 32.WHITE);
display.fillRect(5, 32-bar2, 8, 32.WHITE);
display.display();
display.clearDisplay();
}
else{
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(45,0);
display.println("READY");
display.setCursor(18,8);
display.print("Remote bat:");
display.print(rbatt);
display.println("% ")
display.setCursor(18,16);
display.print("=");

display.print(sbatt);

display.println("% ")

int bar = rbatt/3.125;
int bar2 = sbatt/3.125;
display.fillRect(115, 32-bar, 8, 32.WHITE);
display.fillRect(5, 32-bar2, 8, 32.WHITE);
if (rbatt < 15){
    display.setCursor(18,24);
    display.println("CHARGE REMOTE!");
}
display.display();
display.clearDisplay();
}
}
else{
display.clearDisplay();
display.setTextSize(2);
display.setTextColor(WHITE);
display.setCursor(16,10);
display.print("BREAKING");

int bar = rbatt/3.125;
int bar2 = sbatt/3.125;
display.fillRect(115, 32-bar, 8, 32.WHITE);
display.fillRect(5, 32-bar2, 8, 32.WHITE);
display.display();
display.clearDisplay();
}
t.update();
strip.show();

Figure 14: Arduino code for the controller (3/4).
void batt1board()
{
    int sc = longbatt - 83;
    int di = 100*sc/17;
    sbatt = di;
}

void battMeasure()
{
    spn = analogRead(analogPin3);
    spanning = spn*5/1024;
    just = spanning - 3.5;
    rbatt = just*100/0.7;
}

void updatePixel()
{
    strip.setPixelColor(0,350-s,50,60);
    if (s>149) {
        strip.setPixelColor(1,250-s,20,40);
        if (s>168) {
            strip.setPixelColor(2,20,s+48,210-s);
            if (s>188) {
                strip.setPixelColor(3,10,255,5);
            } else {
                strip.setPixelColor(3,0,0,0);
            }
        } else {
            strip.setPixelColor(2,0,0,0);
            strip.setPixelColor(3,0,0,0);
        }
    } else {
        strip.setPixelColor(2,0,0,0);
        strip.setPixelColor(3,0,0,0);
    }
    else {
        strip.setPixelColor(1,0,0,0);
        strip.setPixelColor(2,0,0,0);
        strip.setPixelColor(3,0,0,0);
    }
}
```c
#include <SoftwareSerial.h>
#include <Adafruit_NeoPixel.h>
SoftwareSerial mySerial(11, 10); // RX, TX
int state;
int newValue;
int x;
int time1;
int time2;
int t;
#define PWM 9
#define LED 8
#define analogPin 0
int ledstrip;
Adafruit_NeoPixel strip = Adafruit_NeoPixel(10, LED, NEO_GRB + NEO_KHZ800);

void setup(){
    Serial.begin(9600);
    mySerial.begin(115200);
    pinMode(PWM, OUTPUT);
    newValue = 130;
    strip.begin();
    strip.show();
    analogWrite(PWM, 0);
    delay(2000);
}

void loop(){
    float val = analogRead(analogPin);
    int batteri = (((val*5)/1024)-4.15)/0.0085;

    if (mySerial.available()) {
        state = mySerial.read();
        mySerial.write(batteri);
        Serial.println(batteri);
        time1 = millis();
        ledstrip = map(state, 89, 207, 0, 255);
        for (int j=0; j<10; j++){
            strip.setPixelColor(j, 255-ledstrip, ledstrip, 0);
            strip.show();
        }
    }
```

Figure 16: Arduino code for the board (1/2).
```c
x = state - newValue;
if (x>1 && 88 < newValue && newValue <= 129) {
    newValue = 130;
    t = 1;
}
else if (x>1 && 119 < newValue && newValue < 208) {
    newValue = newValue + 1;
    t = t*0.4;
}
else if (x<3) {
    newValue = newValue - 4;
    t = 1;
}
else if (x<2) {
    newValue = newValue - 3;
    t = 1;
}
else if (x<1) {
    newValue = newValue - 2;
    t = 1;
}
else if (x==1) {
    newValue = newValue - 1;
    t = 1;
}
else {
    newValue = state;
    t = 1;
}
}

delay(10);
time2 = millis();
if (time2 - time1 > 150) {
    newValue = 130;
    for (int j=0; j<10; j++){
        strip.setPixelColor(j, 255, 0, 0);
        strip.show();
    }
}
for(int n = 0; n<t; n++) {
    analogWrite(PWM, newValue);
}
```

Figure 17: Arduino code for the board (2/2).