The Teabagger - An Automatic Tea Steeper

Mille Mei Zhen Loo
Email: milo@itu.dk

Adrian Valdemar Borup
Email: adbo@itu.dk

Mads Cornelius Hansen
Email: coha@itu.dk

I. Introduction

[Section is written by coha]

As tea enthusiasts, we recognise the need for a device that ensures the precise brewing of tea. The process of brewing tea involves several critical factors, such as the water temperature and the steeping duration. Managing these variables can be challenging, particularly in the morning when time and attention are limited. Hence, the development of a sophisticated brewer capable of executing these specific instructions autonomously is essential. This device would not only guarantee the perfect cup of tea but also free us to attend to other important tasks.

II. Background

[Section is written by milo and adbo]

Currently, the selection of automatic tea steepers is limited. We have identified two types of automatic tea steepers, the first one being a kettle that lowers a basket of tea leafs into water of a specified temperature, and lifting the basket out of the water when the tea is done. A device like this will cost around 1000 DKK, and requires the user to brew an entire kettle at once[1]. The second type of tea steeper is a type of thermos, where the user by the press of a button can release water from a tea steeping compartment into a glass that comes with the product[2].

There are a lot of coffee machines that allows the user to brew a single cup of coffee[3], however this is not the case with automatic tea steepers, and if they do allow the user to brew a single cup, the user is typically unable to use a cup they already own.

Furthermore, the design for The Teabagger is inspired by the action of “teabagging”. In multiplayer video games, “teabagging” is considered a provocative act, where a player repeatedly squats on top of a defeated opponent in order to taunt them. The name stems from the similarity of the player’s scrotum being repeatedly inserted into the opponent’s virtual mouth and the act of dipping a tea bag in a cup of water.

III. Requirements

[Section is written by milo]

- Measure the temperature of fluid (0C - 100C)
- Measure time
- Lower and lift a tea bag
- Resembles a person teabagging

Fig. 1. The fully assembled machine

IV. Description

A. User flow *[Section is written by milo]*

The user first places a mug with hot water on the sieve with the cup underneath, and turns on the device. The user is greeted with the welcome message: ”Welcome to The Teabagger” on the LCD. The user is then prompted to select the temperature, at which the tea should be steeped at, as the LCD is blinking the word ”Temp”. The temperature can be set to any value between 0 and 100 degrees Celsius. The user makes this selection by turning the rotary encoder, and confirms their selection by pressing the rotary encoder. The user is then prompted to select the time for steeping, as the LCD is blinking the word ”Time”. The time for steeping can be set to anything between 0 and 99 minutes. Similarly the user selects the time by turning the rotary encoder, and confirms their selection by pressing the rotary encoder. The temperature is then actively measured from the thermistor. Since the thermistor needs to heat up, the robot waits until the temperature from the thermistor doesn’t increase. If the temperature isn’t increasing but the water is still hotter than the wanted temperature, the robot waits until the water is equal or lower than the wanted temperature. This implies that if the user has poured water that does not reach the indicated temperature, the robot will steep the teabag at the current

temperature. When the water reaches the wanted temperature, the figure will lower the teabag into the tea, and a timer will start. When the timer reaches 0, the robot will lift the teabag out of the tea, and the LCD will print the text ”Your tea has been teabagged”, indicating the tea is ready for consumption.

B. Mechanics *[Section is written by adbo]*

The mechanics of the tea steeper consists of being able to dip a teabag in a cup and remove it once the tea has steeped. The model of a person is positioned above the cup, with the teabag attached to their crotch. To dip the teabag, we simulate the person teabagging, which brings his crotch closer to the cup, and thus brings the teabag into the cup. To remove the teabag, we simulate the person standing back up.

The mechanism is split into two components: the person performing a squat-looking motion and the underlying motor-driven lifting component.

1) Squat mechanism: [Design and implementation was a shared effort by adbo and coha. Section is written by adbo.]

The idea behind the squat mechanism is to have the man’s feet locked in place and put cylindrical joints in the man’s ankles, knees, and crotch. That way, the thighs are always connected to the man’s crotch, and the lower legs are always connected to the feet. But the knees, which connect the thigh and lower leg, are free to move around. Thus, when the torso moves up and down, the thighs and lower legs move in a way that looks like knee extension and flexion, respectively. Figure 2 shows a sketch of the joints and the motion.

However, allowing free rotation around the knee joint can result in the knees caving in. This happens when the legs are extended and the man has to go down into a squat again, but nothing stops the knees from rotating inwards instead of outwards. To see a visual description of the problem, see Figure 2.

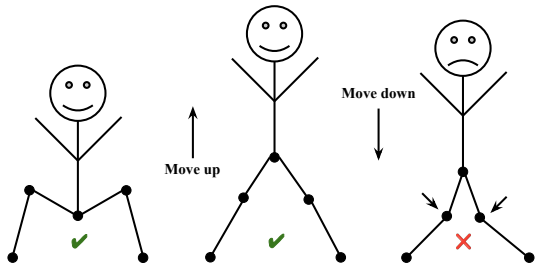

Fig. 2. Demonstration of knees caving in due to too much rotational freedom. Cylindrical joints are highlighted as solid circles.

To solve this, we designed a guide to constrain the motion of the joint in the knee. This mechanism is illustrated and explained in Figure 3. Using this constraint, we reduce the degrees of freedom from 2 to 1 (under the assumption that the crotch and torso can only move vertically and the feet cannot move at all). As such, the position of the torso has a one-to-one correspondence to the state of legs.

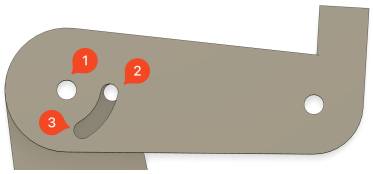

Fig. 3. The mechanism constraining motion of the knee joint. Label 1 shows the cylindrical joint of the knee. Label 2 shows a hole in the thigh, through which a bolt is mounted. The cut-out in the lower leg (spanning from label 2 to 3) constrains the motion of the knee joint by having the bolt collide with either end point. In turn, this limits how much the leg can extend and flex.

For the implementation, we opted for simplicity and use bolts with self-locking nuts as the cylindrical joints. By not tightening the bolts completely, the parts can rotate around the bolt easily—and by using a self-locking nut, we ensure that the bolt doesn’t come too loose over time.

We made a few iterations of the squatting mechanism (because of limbs that got in the way, improved stability, and functionality) as documented in Appendix C-C on Figure 19, 20, and 21.

2) Lifting mechanism: [Design and implementation by adbo. Section is written by adbo.]

To drive the squatting motion, we raise and lower the man’s torso. To do this, we use a belt and pulley mechanism driven by a stepper motor. The man’s torso is mounted onto the belt, such that the torso’s vertical position is controlled by the stepper motor—thus, the stepper motor transitively controls the squatting motion. The pulleys are attached to a pillar structure, standing tall on the base plate. The pillar also supports the motion of the man; the man is mounted directly to the belt, but the mount also wraps around the pillar, acting as a linear guide along the pillar. The described components are shown in an assembly on Figure 4.

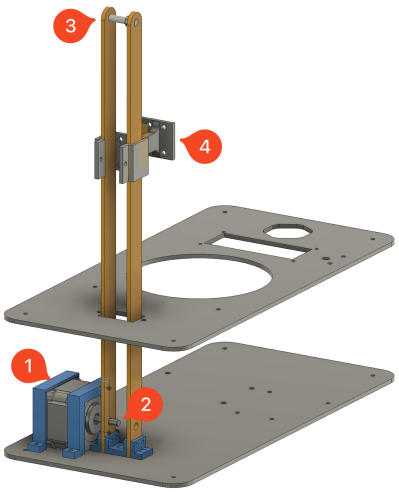

Fig. 4. The components of the belt and pulley mechanism for driving the squatting motion. Label 1 shows the stepper motor. Label 2 shows where the stepper motor connects to the driving pulley (not included in the model). Label 3 shows where the free-spinning driven pulley is mounted. The belt, not shown in the model, wraps around the two aforementioned pulleys. Label 4 shows the mount that connects the belt and the man.

With respect to generating linear motion, we have considered belt drive (which we ended up using), lead screws, and other more complex hand-crafted ideas, which we do not cover here. We went with a belt drive because it was straight-forward to prototype and assemble and it fulfilled our needs, but we will discuss this choice shortly.

To attach our open-ended belt to the mechanism, we used a clamp that can be tightened with a screw (found online [4]). That way, the belt is easy to attach but can still be put under tension afterwards.

3) Alternatives: [Sub-section is written by milo and adbo] The lifting mechanism currently consists of 2 pillars acting as structural elements and a belt to create power transmission between the stepper motor and the attachment to the person. An alternative implementation, that would reduce the number of parts, would use a lead screw instead. The lead screw would both act as a structural element, and as power transmission as it can use the turning motion from the stepper motor, and turn it into a linear motion used for lifting and lowering the person. Furthermore, the self-locking property of the lead screw, would allow the person to securely be in both a standing and squatting position even when the machine is turned off. Additionally, the linear guide we designed for the belt mechanism has rather high friction against the pillar. Using a lead screw, this could be mitigated since the man could be directly attached with a lead screw nut. However, the belt drive was more easily available to us, and it allows us to easily obtain a high-speed squatting motion. But in future iterations of the product, we would opt for a lead screw to simplify the mechanics while making the mechanism more stable.

To translate rotation into linear motion, we initially considered a servo motor, but ended up using a stepper motor. The benefits of using a servo is that the state of the motor is consistent. Even after turning the machine on and off, the electronics can tell the exact the position of the model of the man. However, the distance that the man needs to move necessitates a gearbox with the servo motor. Due to this and concerns about achieving enough torque at our desired speed, we opted for the stepper motor. With the stepper motor, we were certain that enough torque would be provided, and it removes the need for a gearbox. But we also lose the statefulness of the servo. Alternatively, we could add switches to the top and bottom positions to know when either position has been reached. For future iterations, we would measure the force required to run the mechanism. Based on that knowledge, we could decide if a small servo motor available from the lab can provide enough torque to drive the mechanism with a gearbox.

C. Electronics

The machine includes the following electronic components:

- An LCD, to display water temperature and remaining time.
- A thermistor, to measure the water temperature.
- An arduino to send and recieve signals from the different electronics
- A stepper motor to generate motion

1) 16x2 Character LCD: [Design and implementation was a shared effort by milo and coha. Section is written by coha]

In order to display information related to the tea brewing we used a Spark-fun 16x2 Character LCD. The screen displays remaining time, temperature and when the brewing is done. To control the the LCD screen we used the pins as seen in Figure D. Pin 1 is the ground, pin 2 is VCC, pin 3 takes 0-5V and controls the LCD screen lights and is controled by an potentiometer pin 4 lets the arduino switch between the Command and Data Register. pin 5 switches the LCD between read and write, however since we are only interested in writing to the LCD, it is connected to ground. pin 6 control pin for the LCD MCU. pin 10-14 are data pin allowing us to send data to LCD pin 15 and 16 is VCC and Ground for the LED illuminating the LCD, this is also why we have an resistor to lower the back light. We found an guide for the software and set up at [5] and an explanation for the pins at display data sheet [6]

2) DS18B20 temperature sensor: [Design and implementation was a shared effort by milo and coha. Section is written by coha]

To measure the temperature of the tea water we used DS18B20 temperature sensor, which is a water proof sensor that can measure temperatures between -55C to +125C. The sensor have 3 pins **GND**, **DQ** and **VDD**. The sensor have 2 modes but we use it in normal mode which is shown in Figure D

- **VDD** needs be supplied +3V to +5.5V.
- **DQ** is concerted to an arduino pin and according the the datasheet[7] to a 4.7k resistor going to an +3V to +5.5V power supply.

3) Rotary encoder: [Design and implementation by milo. Section is written by coha and milo]

To make the user experience better we used an Rotary encoder to allow user to adjust settings such as dip temperature and time. The reason for using a rotary encoder was the multiple means of interaction, those being turning and pressing the knob. This means we need to use all 5 pins **CLK, DT, SW, VDD** and **GND** is [8] As show in Figure D we had pin **CLK**, **DT** and **SW** connected to different Arduino pins, **VDD** to +5V and **GND** to ground. By using the Arduino to read the changing state of **CLK** and **DT** we can figure out the direction the dial is turned and by how much, this is possible because they have the same frequency but are not synchronised in phase changes as shown in Figure 5. **SW** works like any switch or button by letting current pass when the button is pressed down.

4) Stepper motor [Design and implementation by adbo. Section is written by adbo.]: To run the belt drive, we use a stepper motor to turn rotation into linear motion. As such, our electronics include an external stepper motor driver (A4988), through which we can control the motor from the Arduino. Here, we cover the circuit by going through important pins of the stepper motor driver.

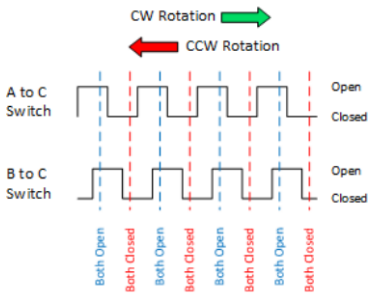

Fig. 5. This shows how pins CLK (A) and DT (B) on a Rotary Encoder change state depending on the direction the dial is turned. Since C is GND, it is possible to measure a current from pins CLK (A) and DT (B) when their phases are open. Image taken from [8]. **CW** clockwise, **CCW** counterclockwise

The step direction is controlled by the `DIR` pin. Depending on the `HIGH` or `LOW` signal being written to this pin, the stepper motor will turn in the corresponding direction.

Similarly, the `STEP` pin is used to control an amount of rotation. For each pulse sent via this pin, the stepper performs a step in whichever direction is specified by the `DIR` pin [9].

The pins `1A` and `1B` were wired to one of the motor's coil pairs, and `2A` and `2B` were wired to the other coil pair.

Finally, the driver requires power sources with two different voltages: one voltage for the stepper motor itself, and another voltage for logic. The logic voltage pin, `VDD`, receives power from the Arduino's 5V pin. The motor power is connected to the 12V power supply via `VMOT`, as it requires a voltage between 8-35V.

In some cases, even at 12V, LC voltage spikes exceeding 35V can occur, which is above the driver's maximum voltage rating [9]. To protect the driver from such voltage spikes, we have put a 100 µF electrolytic capacitor across the motor power.

Additionally, the driver provides a potentiometer to set a maximum current limit. In our case, we do not need to draw a high current, so we limited it to approximately 0.5 A.

Figure 23 in Appendix D shows the schematic.

5) Arduino UNO R3: [Design and implementation by all. Section is written by coha.] We used a single Arduino to be our central hub responsible for reading or sending signals to all our electronic components. On the Arduino we use several pins

- **VIN** needs be supplied +6V to +20V[10].
- **GND** is connected to ground
- **5V** - we use the 5V pin to supply power to most of our electronics.
- **Pin 2-13** we use every pin from 2 to 13 to send or receive signals from all our electronic parts. see section D

D. Software [Design and implementation by milo. Section written by milo]

Since the project has a few electronic components that need to communicate, we gave each component a designated c++ library with a header file to separate code, and only make relevant code visible to the main .ino-file. The main .ino-file use these libraries and make them interact in the setup and loop function. The loop function uses a switch statement and an integer called "globalState" to navigate where in the user flow the user is. When a step is completed the globalState is incremented, and the program progresses.

1) Temperature sensor: [Design and implementation was a shared effort by milo and coha. Section is written by coha and milo]

In order to read the signals coming from the temperature sensor we used two software libraries **One Wire library by Paul Stoffregen**[11] and **Dallas Temperature library**[12] which interprets the sensor signals and converts them to degrees in Celsius. Using the temperature values, we can return the measured temperature by a single call to a function, and we can check whether the temperature is increasing. The latter is needed as the thermistor needs to heat up before giving the correct reading of the liquid temperature. This is done by comparing 2 readings 5 seconds apart.

2) 16x2 Character LCD library: [Design and implementation was a shared effort by milo and coha. The section is written by coha]

In order to write to the LCD screen, we used a library **LiquidCrystal** [13] which worked by converting our instructions into signals was sent through the 6 pins in a way such that the LCD MCU would display the text and numbers we wanted[6]. This allowed us to build functions that mainly allowed for a quick display of the live time and temperature information along with the welcome and goodbye text.

3) Stepper Motor library: [Implementation of software is written by adbo. The section is written by adbo.]

Our stepper motor takes 200 steps to do a full revolution. To turn the motor, the library writes a HIGH signal to the stepper motor's step pin for 700 µs and a LOW signal for 700 µs, repeating the cycle 400 times to do two full revolutions. This results in a theoretical speed of 0.56 s to either squat down or stand up, which we deemed to be an appropriate balance between speed and torque (see subsection V-D for torque tests).

4) Rotary encoder library: [Implementation of software is written by milo. Section is written by milo]

The rotary encoder library has 2 functions corresponding to the 2 modes of interaction with the component. One for turning the shaft, and one for pressing the shaft like a button. Due to the way an incremental rotary encoder works, the component constantly outputs information about the angular motion of the shaft and the state of the button. However, we're only interested when the button changes state, and when the angular motion is non-zero. Therefore, this library ensures that program logic is only executed, when there is a change in these values, and not when they are at a specific value. This is done by keeping track of the previous states of the rotary encoder. The library is used by the main file to make progress in the user flow, and to select the wanted time and temperature.

5) Time tracking: [Implementation of software is written by milo. Section is written by milo]

The time tracking logic is the only part of the program that has not been separated into it's own library. However, with more time, this could definitely be a possible improvement of the software. The time tracking is in the main .ino-file, represented by 3 helper functions that get the elapsed time and returns the elapsed time in seconds. It is used by the main program to tell when the timer has ended I.E. when the tea is done steeping.

V. Results/Analysis

[Tests made by adbo. Section is written by adbo.]

To test our machine, we have devised a series of qualitative and quantitative tests to evaluate its performance. First, we covert the qualitative tests, then the quantitative tests.

A. Qualitative test: can it make a cup of tea?

This test is a end-to-end test that encompasses most of the requirements, focusing on the primary feature of the tea steeper: making a cup of tea. If all the following steps are successfully completed in order, we consider the machine to have passed the test.

1) The machine turns on properly.
2) A tea bag can be mounted onto the hook.
3) Adjusting the rotator knob reflects changes to the configured temperature and time on the display.
4) The brew can be started.
5) The machine waits for the water to reach the appropriate temperature, holding the tea bag above the water until then.
6) Once the configured steeping temperature is reached, the machine lowers the tea bag into the cup.
7) The display shows the steep timer counting down from the configured time, while live-updating the temperature too.
8) Once the timer reaches `00:00`, the machine raises the tea bag out of the cup again.
9) Once the cup is removed, the water remaining in the tea bag drips into the drip tray.

Performing the test, we observe all of the criteria successfully being fulfilled. This is shown in a video demo here [14]: https://youtu.be/bb9PU6Fm0go.

B. Qualitative test: does the mechanism resemble a man performing a squat?

Since this is a subjective opinion, we attempted to make it more objective by comparing the model to a picture of a man squatting (see Appendix F). To judge the squatting motion, we recorded a video of the machine repeatedly performing it [15] and used that to compare. Our own opinion is that it looks like a man squatting, as the relative position of the torso, arms, crotch, knees, and feet are very similar to that of the squat example.

Furthermore, we asked five people external to the group if they would call the motion a "squat". All five responded "yes".

C. Quantitative test: consistency of linear motion after prolonged use

We want to ensure that, even after prolonged use with many cups of tea brewed, the lifting mechanism remains consistent. That is, does the belt drive move the man's torso the same positions consistently over time, or does it drift?

To test this, we programmed the machine to perform the squatting motion 1000 times (simulating 1000 cups of tea), as documented by video [15]. Before starting the 1000 repetitions, we mark the position of the linear guide. After the repetitions finish, we measure the distance between the guide's new position and the old position's marker.

As shown on Figure 6, the guide is essentially in the same spot after the 1000 repetitions. We could not measure a difference in distance using a caliper due to the difference being too small (at most one millimeter).

An improved test could run the same experiment, but repeat it for 100,000 repetitions, which should take between 1 or 2 days to finish.

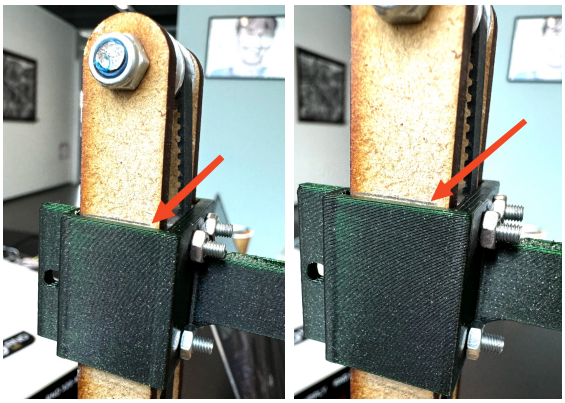

Fig. 6. Pictures of the linear guide's position relative to the baseline pencil marker, before and after the 1000 repetitions, respectively.

D. Quantitative test: maximum weight that can be lifted

We want to test how our mechanism responds to different loads placed on the tea bag hook to see how much load it can handle. Table I shows the different weight loads placed onto the tea bag shaft and whether the machine could repeatedly squat that weight. A weight of 100 g means that the lifting mechanism has to be able to generate enough torque to lift 100 g *on top of* driving the belt and lifting the man (which involves overcoming the high friction of the linear guide, only designed for low weights).

"Partially" squatting means that the belt drive fails to consistently move the torso between the same top and bottom positions, but it does lift the item repeatedly.

With these results in mind, we rate the machine to be able to lift approximately 400 g of additional weight. This corresponds to the weight of 20 fully-soaked tea bags or 80 dry tea bags. For the intended purpose, this is plenty.

If we wanted to improve the load capacity, one option would be to make an improved linear guide with less friction on the

TABLE I

Weight	Able to lift weight	Item
5 g	Yes	Dry tea bag
20 g	Yes	Wet tea bag
75 g	Yes	Piece of plastic
200 g	Yes	Small size stepper motor
275 g	Yes	Combination of previous items
300 g	Yes	Medium size DC motor
345 g	Yes	Full soda can
375 g	Yes	Combination of previous items
420 g	Partially	Combination of previous items
475 g	No	Medium size stepper motor
500 g	No	Combination of previous items

main pillar. An even stronger solution would be to use a lead screw instead of belt drive.

E. Quantitative test: accuracy of steep time

We want to test, given a specific steep time, how accurately the machine can steep the tea for that duration. That is, if it claims to steep for 7 min, how close is the actual steep time to that? We want to investigate this because the time that tea is steeped for has a significant impact on its flavour [16].

The experimental setup here is that we, for varying steep times, have a stopwatch on the side. We start the stopwatch at the same time as we start the brew and stop the stopwatch once the steep timer finishes. We then compare the time difference.

TABLE II
RESULTS OF THE TIME ACCURACY EXPERIMENT

Steep time	Stopwatch result	Time difference
3 min	3 min 0 s	0 s
5 min	5 min 0 s	0 s
7 min	7 min 1 s	1 s

These results indicate that the steep timer is accurate to within a second on up to 7 min steep times. The time difference is most likely caused by human error when starting the stopwatch.

The squatting motion itself to dip the tea bag and raise the bag again also takes roughly a second. We conclude that the margin of error of around a second for the steep time is acceptable for steeping tea.

F. Quantitative test: accuracy of temperature

Alongside different steep times, different teas require different temperatures, which impact the final flavour of the tea noticeably [16]. To perform this experiment, we would use an external trustworthy thermometer and compare its reading to what is output on the display.

Due to a lack of a thermometer, we did not have time to perform this experiment.

According to the temperature sensor's data sheet, it is accurate to within 0.5 °C in the range from −10 °C to 85 °C and accurate to within 1 °C in the range from −30 °C to 100 °C [17].

Tea is commonly steeped between 80 °C and 100 °C, where the user can expect a ±1 °C error, which we deem acceptable since the water cools down by multiple degrees during the steeping process anyways.

VI. DISCUSSION

[Section is written by milo and coha]

Our tea brewer excels in brewing a cup of tea with style and offers multiple options for users to customize their brewing experience. It's slim design fits perfectly on a kitchen table. However, there are some areas for improvement.

Firstly, our lifting mechanism could become more stable and reliable by replacing it with a spinning lead screw as described in subsubsection IV-B3. Although this would require some modifications, it would result in more stable and precise movements. Additionally, we need to widen or enhance the support pillars, as we discovered they flex under heavy loads and quick movements.

A further needed improvement of the lifting mechanism is that the tea steeper is currently only able to steep tea for cups that fall into a specific height requirement, namely 8-10 cm. A workaround to be to make the legs of the person longer, thereby increasing the squatted distance. A possibly better workaround while still fulfilling all of the requirements would be enabling the phallic attachment to spin, thereby winding the string of the teabag around the protruding cylinder. This would use the string of the teabag to extend the range in which the tea can be lowered.

Finally, if this product were to be manufactured on a large scale, we would need to design a new water resistant casing to cover all electronics, both to improve visuals for customers and to make sure that in case of spills the electronics would be protected. Another enhancement would be to incorporate a small speaker or noise maker to signal when the tea is finished. Additionally we could design some sort of automatically disposal of used tea bags, eliminating the need for the user to do it each time. A last improvement would be to find a mechanism that automatically lowers temperature sensor when a cup is placed and lifts it again the the tea is done.

[A more exhaustive list of the responsibilities of each group member during the project can be found in Appendix A]

REFERENCES

[1] Sage, "The tea maker compact," https://www.imerco.dk/sage-the-tea-maker-compact-1-liter-1600-watt-graa?id=100418724, 2024, accessed: 30-Jul-2024.

[2] Zens, "Modern lazy glass tea infuser teapot: 3-in-1 smart tea maker for loose leaf tea steeper," https://zensliving.com/products/zens-modern-lazy-glass-tea-infuser-teapot-3-in-1-smart-tea-maker-for-loose-leaf-tea-steeper, 2024, accessed: 30-Jul-2024.

[3] Nespresso, "Citiz," https://www.nespresso.com/dk/da/order/machines/original/citiz-hvid-kaffemaskine, 2024, accessed: 31-Jul-2024.

[4] AndyFromSpace, "GT2 Inline Belt connector clamp," 2018. [Online]. Available: https://www.thingiverse.com/thing:2802253

[5] Arduino, "Lcd displays," https://docs.arduino.cc/learn/electronics/lcd-displays/, 2024, accessed: 29-Jul-2024.

[6] A. Raj, "16x2 lcd display module - pinout & datasheet," https://circuitdigest.com/article/16x2-lcd-display-module-pinout-datasheet, 2015, accessed: 29-Jul-2024.

[7] M. Integrated, "Ds18b20: Programmable resolution 1-wire digital thermometer," https://cdn.sparkfun.com/datasheets/Sensors/Temp/DS18B20.pdf, 2008, accessed: 29-Jul-2024.
[8] H. Tech, "Rotary encoder module datasheet," https://www.handsontec.com/dataspecs/module/Rotary%20Encoder.pdf, 2024, accessed: 29-Jul-2024.
[9] Pololu, "A4988 Stepper Motor Driver Carrier," visited Jul. 31st, 2024. [Online]. Available: https://www.pololu.com/product/1182
[10] Arduino, "Arduino uno rev3 datasheet," https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf, 2024, accessed: 30-Jul-2024.
[11] P. Stoffregen, "Onewire library," https://github.com/PaulStoffregen/OneWire, 2024, accessed: 30-Jul-2024.
[12] M. Burton, "Arduino-temperature-control-library," https://github.com/milesburton/Arduino-Temperature-Control-Library, 2024, accessed: 30-Jul-2024.
[13] Arduino Libraries, "Liquidcrystal library," https://github.com/arduino-libraries/LiquidCrystal, 2024, accessed: 30-Jul-2024.
[14] The authors of this report, "End-to-end demo of Tea Bagger," 2024. [Online]. Available: https://youtu.be/bb9PU6Fm0go
[15] ——, "Demo of Tea Bagger repeatedly performing a squat," 2024. [Online]. Available: https://youtube.com/shorts/ZlcbchasEHk?si=HInbkdEvP1ZtpG5W
[16] hotwatertaps.com, "What Temperature Should You Brew Your Tea At?" visited Jul. 31st, 2024. [Online]. Available: https://hotwatertaps.com/tea-temperatures/
[17] Maxim Integrated Products, Inc., "Ds18b20: Programmable resolution 1-wire digital thermometer," 2019. [Online]. Available: https://circuitdigest.com/article/16x2-lcd-display-module-pinout-datasheet

APPENDIX A
PROJECT RESPONSIBILITIES BY GROUP MEMBERS

The following table outlines which people were *primarily* responsible for a specific part of the project. For cases with multiple group members, an estimated percentage of distributed work is given.

Part of project	Group members
Squat mechanism	
Mechanism: concept, sketches, design, how-to	adbo
Creating 3D models	coha
Manufacturing: printing, laser-cutting	coha
Assembly	adbo (33%), coha (66%)
Lifting mechanism	
Mechanism: concept, sketches, design, how-to	adbo
Creating 3D models	adbo
Manufacturing: printing, laser-cutting	adbo
Assembly	adbo
Base plate	
Base plate 3D modeling	33%coha, 66%milo
Drip tray modeling and manufacturing	milo
Electronics (incl. schematics)	
Rotary encoder	milo
LCD display	coha 80%, milo 20%
Stepper motor	adbo
Thermometer	coha, milo
Combining circuits	milo
Software	
General logic	milo
Rotary encoder	milo
LCD display	coha 20%, milo 80%
Stepper motor library	adbo
Thermometer	coha 33%, milo 66%
Other	
Test setup and evaluation	adbo
Report writing	Documented in sections

APPENDIX B
BILL OF MATERIALS (BOM)

A. Electronics

TABLE III
BOM FOR ELECTRONICS SHARED BY ALL COMPONENTS
[Table created by everyone]

Qty.	Item
1x	Arduino Uno
2x	Breadboard
1x	DC power supply (outputting 12 V, limited to 0.6 A)

TABLE IV
BOM FOR THE STEPPER MOTOR ELECTRONICS *[Table created by adbo]*

Qty.	Item
1x	Stepper motor (bipolar), model 17HS13-0404S
1x	Polulu A4988 Stepper Motor Driver
1x	Electrolytic capacitor (50 V, 100 µF)
1x	Mini breadboard (with 34x 5-hole terminal strips)
11x	Jumper wire
	Item (power source)
	12V DC power source (refer to Table III)
	5V DC power source (via Arduino 5V pin, refer to Table III)

TABLE V
BOM FOR THE USER INTERFACE ELECTRONICS *[Table created by milo]*

Qty.	Item
1x	DS18B20 temperature sensor
1x	Rotary encoder
1x	4.7k resistor
1x	sparkfun LCD screen

B. Mechanics

TABLE VI
BOM FOR THE LIFTING MECHANISM *[Table created by adbo]*

Qty.	Item (belt component)
1x	GT2 belt: 623 cm long, 2 mm pitch, 6 mm width, open-ended, rubber
1x	GT2 idler pulley: with bearings, 20 teeth, 5mm bore, 6mm belt width
1x	GT2 timing pulley: with set screw, 20 teeth, 5mm bore, 6mm belt width
1x	GT2 belt connector clamp by AndyFromSpace [4]
1x	M3 bolt (12mm) with M3 nut to fasten clamp
	Item (pillar component)
2x	Pillar side wall laser-cut from 3mm MDF (see Figure 11)
1x	M5 bolt (30mm) with M5 nut to hold tighten the top of pillar
2x	Pillar spacer (see Figure 15)
2x	Outer pillar mount (see Figure 13)
1x	Inner pillar mount (see Figure 13)
3x	M3 socket countersunk fastener (10 mm) to fasten pillar mounts to base plate
4x	M3 bolts (10mm) to fasten pillar walls to pillar mounts
7x	M3 nuts for corresponding bolts
	Item (stepper motor component)
1x	Stepper motor rear mount with hole (see Figure 13)
1x	Stepper motor front mount without hole (see Figure 13)
4x	M3 bolts (10mm) to mount stepper motor to base plate
4x	M3 nuts for corresponding bolts
1x	Stepper motor circuit (refer to the electronics BOM in Table V)

TABLE VII
BOM FOR THE MODEL OF THE SQUATTING MAN *[Table created by coha]*

Qty.	Item (body of the man)
1x	man body
2x	man leg
2x	man calf
1x	tea hook
4x	M3 bolts (12mm) to connect the mans legs to calf's
1x	M3 bolts (16mm) to connect the mans legs, body and tea hook
5x	M3 nyloc nuts
	Item (leg supporting pillars)
2x	foot pillar
2x	pillar mount
2x	foot pillar
6x	M3 bolts (12mm) to mount the (foot pillars) to (pillar mounts) and to the mans calf's
2x	M3 countersunk bolts (12mm) to mount (pillar mounts) to (base plate top)
2x	M3 nyloc nuts
6x	M3 nuts
	Item (man to belt mount)
1x	gt2 belt clamp
1x	gt2 clamp to man
6x	M3 bolts (12mm) to tighten the two clamps together and mount (gt2 clamp to man) to (man body)
2x	M3 countersunk bolts (12mm) to mount (gt2 clamp to man) to (man body)
8x	M3 nuts

TABLE VIII
BOM FOR THE MODEL OF THE BASE PLATE *[Table created by coha]*

Qty.	Item (base plate)
1x	base plate top
1x	base plate bottom
8x	3M (30mm) hex spacer, 2 spaces are joined in each of the 4 corners between base plate top and button.)
4x	m2 bolt (20mm) to mount LCD screen
4x	m2 nuts
12x	m3 bolts (12mm) to screw the spacers and base plate top together, mouth breadboard holder and mount the arduino.
2x	m3 bolts (20mm) to mount Rotary encoder
14x	m3 nuts
1x	breadboard holder
1x	drip tray
1x	drip sieve

APPENDIX C
TECHNICAL DRAWINGS

A. Full assembly *[Design is a even shared effort from all group members. Section written by adbo.]*

This section showcases the major parts of the combined assembly. However, the following subsections will focus on each mechanism on its own and show more detailed drawings of them.

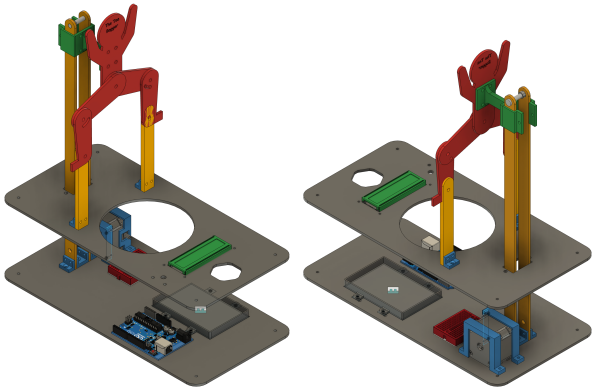

Fig. 7. Important parts of the full assembly. It is missing some components from the real-life assembly, such as hex spacers between the base plates, the middle drip tray, pulleys and belt, wiring, bolts and buts, and a rotator knob for user interaction. Parts have been shaded to more easily distinguish between them.

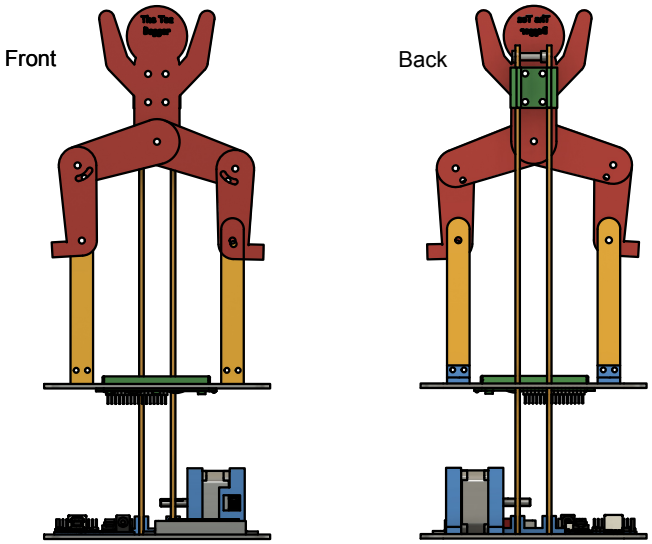

Fig. 8. The full assembly from Figure 7 seen from the front and back.

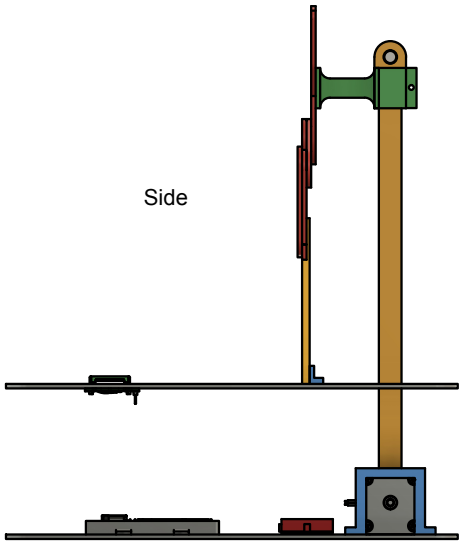

Fig. 9. The full assembly from Figure 7 seen from the side.

Fig. 10. This is the an drip tray and drip sieve intend to slide over the hole in Figure 7, this allows users to place thire cup on the machine and for all spillage to be collected safely underneath.

B. Belt drive lifting mechanism *[Section is written by adbo]*

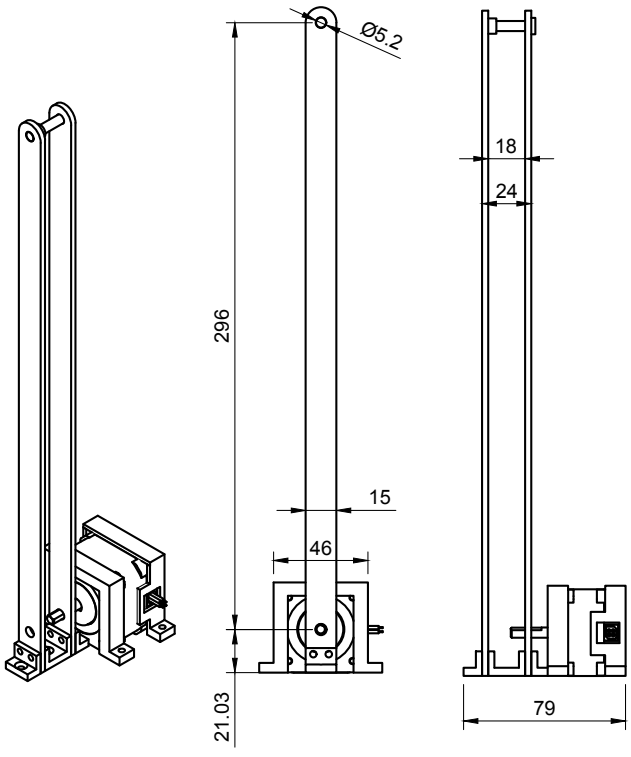

Fig. 11. Technical drawing of the belt drive mechanism assembly on its own with general dimensions added. All numbers are in millimetres. See below figures for more descriptions and model views.

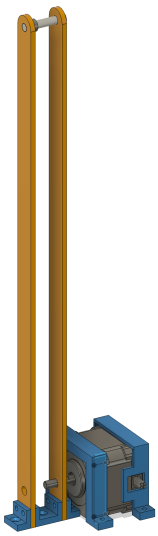

Fig. 12. A coloured 3D model of the belt drive mechanism on its own. Note, the model does not show the pulleys nor the belt. See Figure 13 and Figure 15 for more in-depth pictures. *Disclaimer: the model of the stepper motor (grey) was found online. All other shown parts have been modelled by ourselves.*

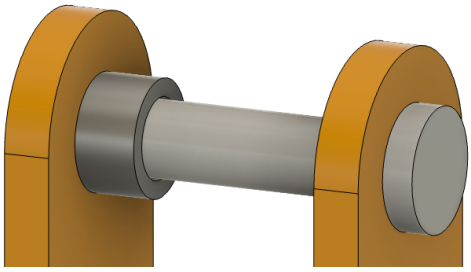

Fig. 14. A zoom-in on the top part of Figure 12, showing the M5 bolt that tightens the top part of the pillar as well as a spacer. In reality, an idler pulley (with bearings) will be in the middle, with spacers on each side, keeping the pillar at the correct width and the pulley in the middle.

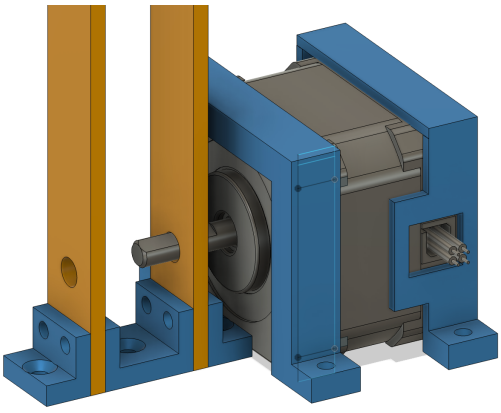

Fig. 13. A zoom-in on the bottom part of Figure 12, showing the pillar side walls, the three mounts in the bottom for the side walls, the stepper motor, and the stepper motor's two mounting brackets. It is also shown how the stepper motor axle sticks through a hole in the side wall, where it can connect to a pulley.

The hole, after being manufactured, was filed a bit larger to account for the fact that the stepper motor shaft is pulled slightly upwards once the belt is put under tension. As such, the shaft would grind against the side wall hole if the hole was not made larger.

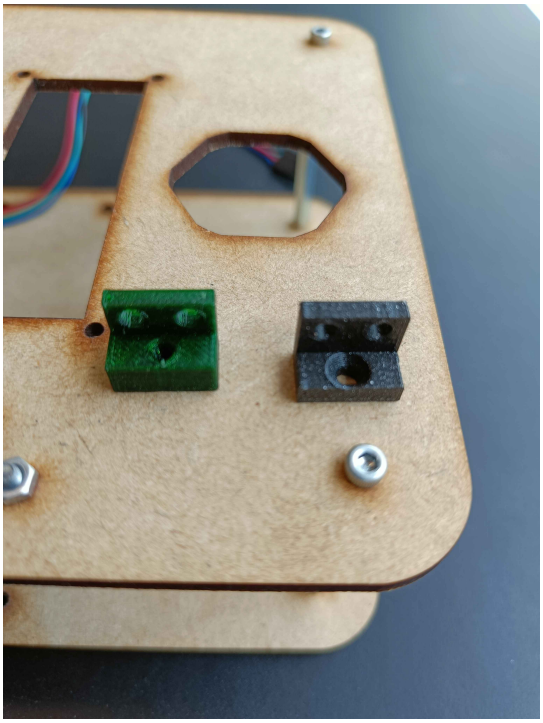

Fig. 15. showing two iterations of the pillar mounting parts. On the left is the first version where bolts would collide because of missing space. On the Right we fixed that by making an indent in the part so all bolts could fit and be fastened

C. Model of man *[Section is written by coha]*

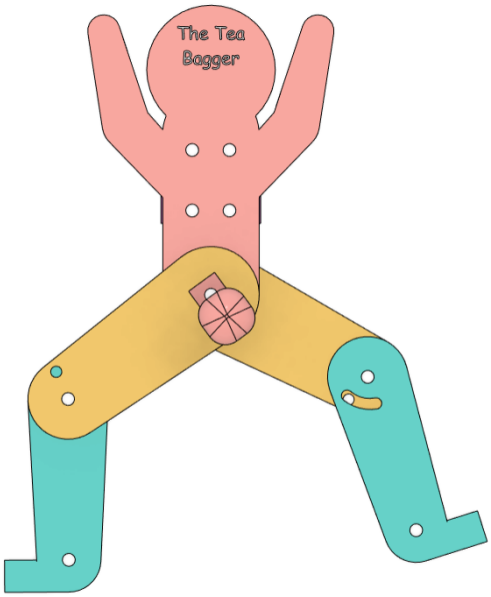

Fig. 16. An overview of the assembly of the man. The man is a grope of parts that are fasten with lose M3 bolts making each limb and part a revolution joint.

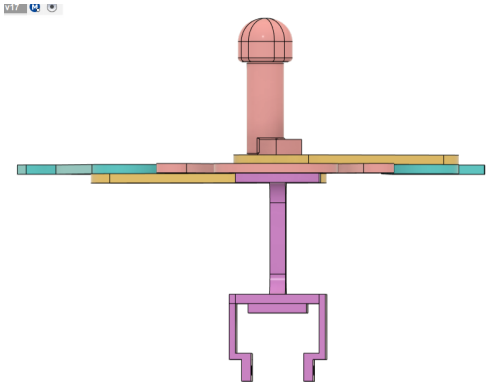

Fig. 18. This is the top view of Figure 16,here we can see how the clamp is intend to go around our tower as support and be attach the the belt in the middle part.

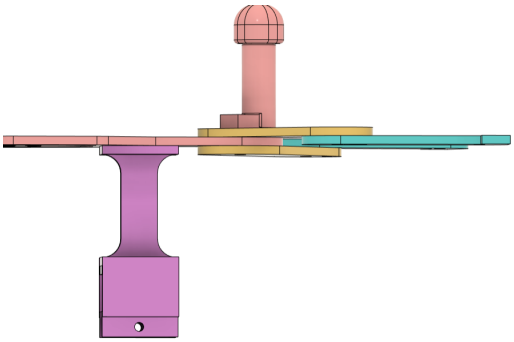

Fig. 17. This is the side view of Figure 16, here we can see how the clamp which attaches the man assembly to the belt is mounted. the clamp is bolt with 2x M3 bolts and nuts in Figure 16 top two holes and in the two holes below with 2x M3 Countersunk Bolt to avoid the legs colliding with the bolts.

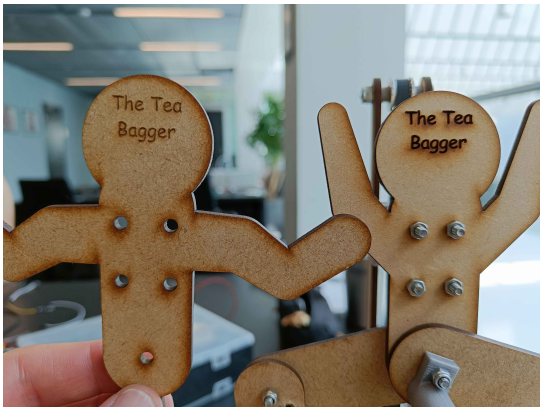

Fig. 19. The man design underwent some changes, as shown by the man on the left we started with his arms down, but they ended up hitting his legs so we changed the man to have his arms in the air as shown on the right

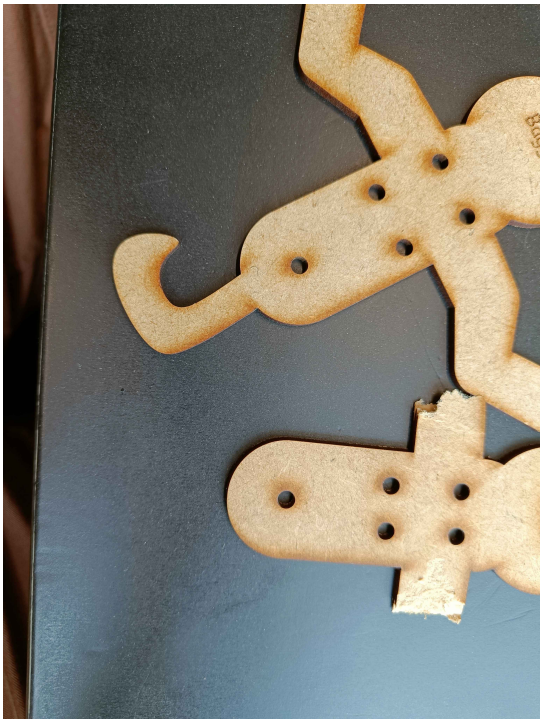

Fig. 20. Our mechanism for holding the tea bag started originally as a hook part for the man body, But it was difficult hang the tea bag so we redesign the hook to be mounted on the man instead.

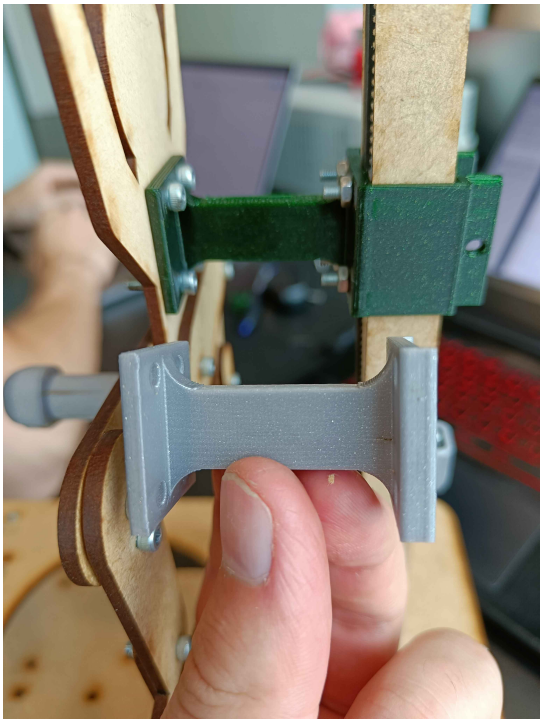

Fig. 21. The original design (the gray clamp) for the clamp mounted to the man was only connected to the man and the belt, but that resulted in the man shaking from side to side and moving a bit vertically. So we redesigned the clamp to also wrap around the pillar for additionally stability and more controlled movement.

APPENDIX D
SCHEMATICS

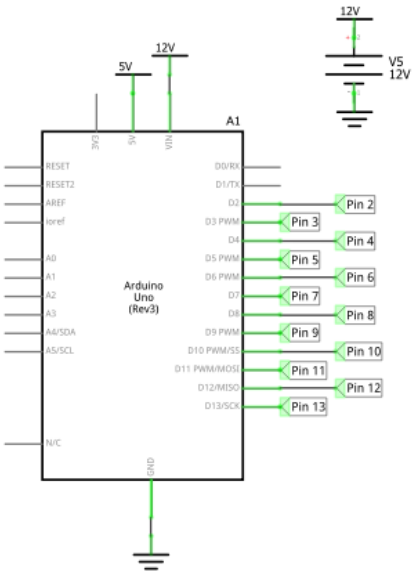

Fig. 22. Arduino component schematic *[Created by adbo and milo]*

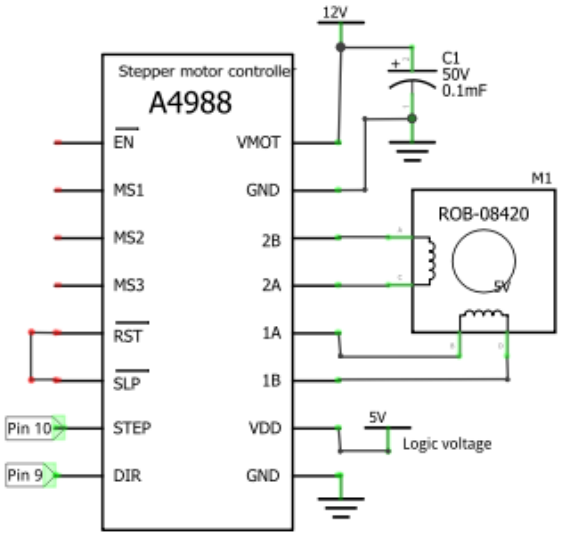

Fig. 23. Stepper motor component schematic *[Created by adbo]*

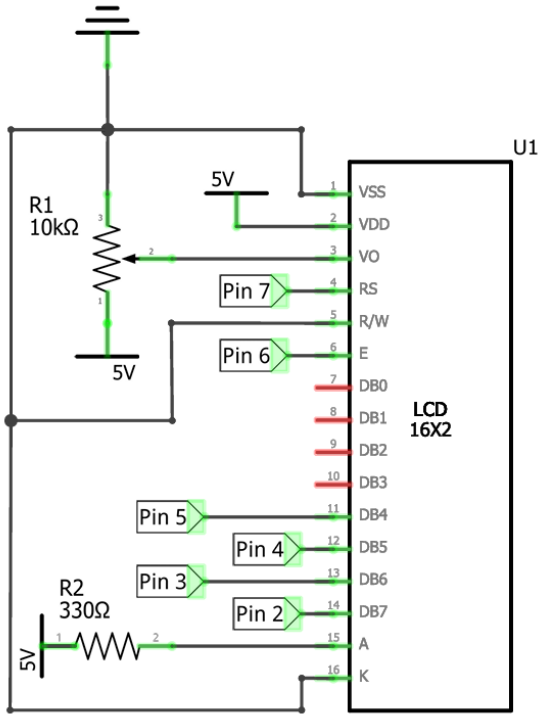

Fig. 24. LCD screen connected to the Arduino *[Created by milo]*

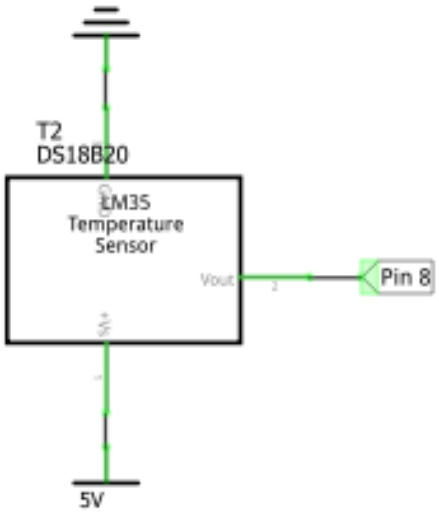

Fig. 26. DS18B20 Temperature sensor *[Created by coha]*

Appendix E
Software documentation

If you think that is necessary to understand the code. This is optional

Appendix F
Example of man squatting

The following picture shows a man squatting. Specifically, the man is in the lower position of an Olympic weightlifting move called “the snatch”.

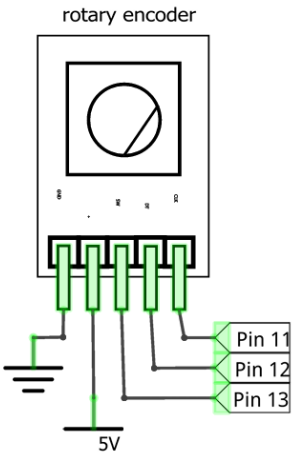

Fig. 25. Rotary encoder connected to the Arduino *[Created by milo]*