Pill distribution machine
End report

Erik Zilliacus 428190
Joakim Högnäsbacka 425339
Casimir Westerholm 476016
Teresia Borgman 428501
Victor Stigzelius 431983
1. Introduction

The main idea for our medicine dispensing machine was to have a user friendly interfaces that communicates with the user and will give out the right amount of medicine. Our product would make the remembering and keeping track of pills easier, as to once it is programmed correctly it will run smoothly until you update the machine. Our machine is mainly meant for elderly people or to be used at hospitals, who on average consume larger quantities of different pills. The product is small enough to be used at home and it doesn’t take a lot of space.

In this report we will explain how we built our machine so that anyone who reads this will know what tools and what equipment we used to make this prototype possible.

2. Electromechanics

The goal was to make the fundamental electromechanical structure of the device as simplistic as possible. We succeeded in shrinking it down to only two motors and one sensor. Essentially, a servo motor controls the pill grabbing and the stepper motor controls which sort of pill to grab.

In the rotation of the containers the angular precision is essential. That is why we decided to use a stepper motor for this function. If we want to go to position n we just have to know what the angular displacement relative to the current angular position and convert that into steps. The motor will always do just that, without any errors. Because of the fundamental structure of a stepper motor, all the possible angular positions are discretely fixed. The risk of losing steps is essentially removed by calculating the maximum possible weight of the rotating module (all containers are filled up with pills) and choosing a motor with a large enough torque to rotate that mass.
The stepper motor we used was the Nema23 (200 steps) with a torque of 125 ounce-inches, or 0.88 Nm. With our estimations this torque was enough and when later testing the motor attached to the load we noticed that it was more than enough; it could be replaced by a smaller and cheaper motor.

In the grabbing of the pill the precision of the motor is less important, but a closed loop feedback is essential for continuously knowing the approximate angular position. This, because we need to know with absolute certainty if the arm is stretched out or not. For example if the stepper starts to rotate when the arm is stretched it would cause the whole system to be stuck and potentially damage the mechanical structure. That is why we chose a servo motor for this function. The DF05BB by DFRobot worked perfectly for this purpose. With a built-in potentiometer and operating voltage 5V this makes it very convenient to control directly with the Arduino.

Getting linear displacement from angular displacement is the basic function of the servo motor in our case. We used a rack and pinion gear setup to achieve this. The image shows the 3D-printed result with arrows showing the direction of the movement.

(The code for the stepper and the servo is in chapter 9.1)

![Image 1: rack and pinion gear](image)

3. Electronics

The electronics of the project is very simple. We have mostly used ready modules that do not need any additional electronics to operate.
As the brain of the device we have used the microcontroller Arduino Mega 2560. For controlling the stepper motor efficiently with the Arduino we have used the EasyDriver v.45 stepper motor driver.

The only additional electronic circuitry we have used is a transistor as a switch between the stepper motor driver and ground. The gate of the transistor is connected to a digital pin of the Arduino. This pin is low during the first milliseconds in the code and then turns on, allowing current to flow through from source to drain. After this the stepper motor is powered and ready to function. This prohibits any voltage spikes during powering on of the device to cause any extra steps being taken by the stepper motor. This was a problem we noticed early on which made the rotating containers to be out of phase.

The Arduino is powered by a 12V DC adapter. This voltage can be used from the Vin-pin and is the voltage the stepper motor operates on, so EasyDriver is connected to Vin.

The servo motor only needs 5V which it conveniently gets from the built-in regulated 5V pin of the Arduino. The SERVOSIG in the schematics is connected to the third wire of the servo from which it can be controlled directly by the Arduino. The servo motor has a built-in potentiometer so its angular position is always known.

4. Software

The goal with our programming module was to make it as user friendly and versatile as possible. The programming was done with Arduino and it is connected to a computer so that the user can communicate with the program through serial monitor. The program will ask the user what time it is during setup and because we didn’t have a clock module, we implemented function millis(); which counts the milliseconds since the start of the program. This will allow us to continuously keep track of the current time by adding the converting the passed milliseconds into minutes and the time the user defined in the setup phase. After time setup the program prints to the user a question whether he/she wants to view the current content of a specific box or insert a new medicine.
If the user chooses to view the current content of the box for the first time it will print to the user that the current box is empty. This is done with the help of a variable for each class, which counts if the class has been initialized further than their zero value. The view option enables the user to view current content of a specific box along with essential information, such as the amount of pills remaining, the remaining days of the course and the amount of leftover pills after the course has ended.

In the insert option it will ask the user what the name of the medicine is, how many pills will be put in the specific box, the length of the course, and how many times said pill will be ingested per day, divided into morning and evening pills. The program will then save the aforementioned input values into a class and will produce 4 different classes, independent of the others, meaning it will create a class for each box and each box could have different types of pills with different specifications. If the user accidentally puts the amount of pills in the machine to be smaller than the needed for the course (Pseudocode: if(amount <(courselength*perDay)){} it will print to the user through serial monitor that the amount filled in to the machine is smaller than needed and will restart the inputting of current box.

The input values that the user feeds through serial monitor, will change the mechanical functionality in matter of how the stepper and servo motor are controlled in the use of outputting pills. The mechanical code applies the parameters from the user interface to determine how the rack and pinion gear will be controlled and the control of the stepper to rotate to specific phase. For example if a box does not have any morning pills the mechanical code will skip said box during its morning output of pills.

(The code for the software can be found in chapter 9.2)
5. 3D-modelling

The construction of our machine was made quite simple. It consists of only 14 different parts of which 11 were 3D-printed. The bottom part is a hollow cylinder with a modelled place for the stepper motor. The middle part is placed on top of the bottom part where it holds up the hanging part where the servo is placed. We made the servo hang down from the middle part so that it wouldn’t rotate as it would if it had been fastened a easier way such as on the stepper motor. On the top of the middle part is the hat with the opening where the pills are filled into the machine. From the stepper motor arises another hollow cylinder where the pill containers are attached. This cylinder has rails on its sides, where the containers are placed. In this way the pieces are easy to take apart and therefore also easy to clean.

The material we used was white PLA, because that was the only type that was available in our printer. We used 670g and in all it took about 50 hours to make the parts, including modelling and printing. If we would start producing these machines we would probably choose another type of plastic, for example ABS, and a more ‘non-hospital’ color. We would also use injection molding instead of 3D-printing to prevent the irregularities caused by the conversion to stl format, and it would also make the production cheaper if they were mass produced.
6. Instruction for use

The first step for the user to do once it has got the prescribed medication is to set-up time to the machine so it can keep track on how many days has past and when it is time for the user to take his/her pills. The system will ask what medicine it is, how many pills, the length of the course and how often a pill must be taken. The user fill in the information given on the medicine box. In this case the user would type in e.g. (What is the name of the medicine?)--> “Burana”, (Amount of pills?) -->“42”, (Course length?)--> “3w” and (Amount of morning pills?)-->”3”, (Amount of evening pills?)-->”1”.

7. Budget

We had estimated a budget for the whole project to be 100€ - 200€ and this was pretty much how it turned out. The total amount we paid for can be seen below:

<table>
<thead>
<tr>
<th>Component</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vacuum Pump</td>
<td>18,34 €</td>
</tr>
<tr>
<td>Stepper motor</td>
<td>30,27 €</td>
</tr>
<tr>
<td>Easy Driver</td>
<td>16,17 €</td>
</tr>
<tr>
<td>Plastics</td>
<td>54,45 €</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>119,23 €</strong></td>
</tr>
</tbody>
</table>

The project would have been more expensive if we did not have some key components in the beginning like the Arduino and servo.
Below is an estimation of what the project would cost if you would buy every component:
The microcontroller is chosen as an Arduino nano, because it would work as well and is cheaper. The only additional thing you would need with this microcontroller is a DC power plug for connecting the 12V to the stepper motor. The stepper motor is a little more robust than it would need to be for this project. Therefore the cost could be further reduced by buying a little weaker and cheaper one.

### 8. Development ideas

What we could still develop with our product is making it an IoT. So that every time the distributor gave out pills it would send a message to the user’s phone/tablet or any other device to remind him/her that it is time to take their medicine.

An other idea was that we could add a weight sensor beneath the pill-cup, and it would notice every time the user takes his/her medicine. If the cup isn’t removed in a while when it should be, it alarms the user again to take the pills.

We also discussed that a bar code reader could be added to the product. This would be a major relief for the user so that he/she doesn’t need to type in what kind of medicine or any other information about the medicine. The results of this means that the user would only poor the pills in the machine and take them when they pop out.

Other ideas were mainly about design. The containers were made of see-through materials glass so that the user could see the containers spinning and keep track of current content of box and in fact see what is happening, which makes everything seem more user friendly.
9. Codes

9.1 Code 1

```cpp
#include <Servo.h>

Servo servo;

int servoPin = 9;       //The pin for controlling the servo
int stepDirectionPin = 2;
int stepPin = 3;

int current = 4;        //Variable for current container
int armOut = 20;        //The angle of the servo when the arm is out
int armIn = 140;        //The angle of the servo when the arm is in
int gate = 53;          //The pin that goes to the gate of the transistor

void setup() {
    pinMode(gate, OUTPUT);
    pinMode(stepDirectionPin, OUTPUT);
    pinMode(stepPin, OUTPUT);
    servo.attach(servoPin);
    digitalWrite(stepDirectionPin, LOW);    //Makes sure that the PINS for the stepper are low in the beginning
    digitalWrite(stepPin, LOW);
    analogWrite(gate, 0);                   //Gate pin LOW, which prohibits any voltage spikes to make the stepper move an extra step when
    analogWrite(gate, 255);                 //This "turns on" the stepper motor. When the gate is HIGH the transistor lets current through
    servo.write(armIn);                     //Makes sure that the arm is in, in the beginning
}

void loop() {
    delay(1000);                           //A small delay before the stepper is turned on
    analogWrite(gate, 255);                //Code for Demonstrating the device
    current = switchContainer(3, current);
    grabPill(3);
    current = switchContainer(4, current);
    grabPill(3);
    current = switchContainer(3, current);
    delay(1000000);
    grabPill(3);
    current = switchContainer(2, current);
    delay(2000);
    current = switchContainer(1, current);
    grabPill(1);
    delay(2000);
}```
current = switchContainer(2, current);

//A function for the servo to take a pill from the current container
void grabPill(int amount) //amount: The amount of pills that should be taken
{
    for(int i = 0; i < amount; i++) //Repeat amount times
        servo.write(armOut); //The servo moves it arm out
        delay(2000); //Waits for a pill to fall into the "hand"
        servo.write(armIn); //The arm moves back in so that the pill falls down.
        delay(1000); //The servo needs a small break
}

//A function for switching current container
int switchContainer(int number, int currentCont) //number: the number of the container to be switched to, currentCont: current container
{
    int container[] = {0, 400, 800, 1200}; //An array for the steps needed to be taken for the corresponding containers
    int stepCount = 0; //A variable for counting the steps taken
    int target = container[number - 1] - container[currentCont - 1]; //calculates the number of steps needed to be taken to reach the target
    int microDelay = 1100; //The delay between each step; determines the angular velocity of the stepper

    //Determines if the stepper needs to rotate counter-clockwise or clockwise and switches the direction pin accordingly
    if(target < 0){
        digitalWrite(stepDirectionPin,LOW);
    }
    if(target > 0){
        digitalWrite(stepDirectionPin,HIGH);
    }
    target = abs(target);

    //Starts to rotate the stepper one microstep at a time (default mode) and counts steps into the stepCount variable until target is reached.
    for(;target > stepCount;stepCount++){
        digitalWrite(stepPin, HIGH);
        delayMicroseconds(microDelay);
        digitalWrite(stepPin, LOW);
        delayMicroseconds(microDelay);
    }
    //Returns the current container so that it can be stored in the master code
    currentCont = number;
    delay(800);
    return currentCont;
}
9.2 Code 2

```cpp
#include <Servo.h>
//the variables underneath are used in the loop but need to stay unreset
int Hr;
int Min;
int Secs;
int Days;
int milliS;
int varE;
int varF;
int varG;
int varH;
int varI;
int viewslot;
int currentlokero = 1;

//Creating a class Lokero that contains each box's information, the system in its current form will create 4 classes
//the class based approach enables flexibility in amount of boxes for further development
class Lokero
{
    String Nimi;
    int maara;
    int maara_paiva;
    int maara_jaljella;
    int paivia_jaljella;
    int ylimaaraisia;
    int paivia;

public:
    int aamu;
    int ilta;

Lokero(String typeString, int amount, int courseLength, int mPill, int ePill, int day)
{
    Nimi = typeString;
    maara = amount;
    maara_paiva = courseLength;
    aamu = mPill;
    ilta = ePill;
    paivia = day;
    maara_jaljella = amount;
}

//The serial monitor will print back to the user his/her input values for parameters, name of medicine, amount of pills, course length,
//amount of morning pills, amount of evening pills.
void testprint()
{
    Serial.print("The name of the medicine: ");
    Serial.println(Nimi);
    Serial.print("The amount of pills: ");
    Serial.println(maara);
    Serial.print("Course length: ");
    Serial.println(maara_paiva);
    Serial.print("The amount of morning pills: ");
    Serial.println(aamu);
    Serial.print("The amount of evening pills: ");
    Serial.println(iltta);
    Serial.println("\n");
}
```
//A function that updates the user about the time bound information about the chosen pill type
void testprintjaljella()
{
    Serial.print("Amount of pills left: ");
    Serial.println(maara_jaljella);
    Serial.print("Amount of days left: ");
    Serial.println(maara_paiva);
    Serial.print("Excessive pills after course has ended: ");
    Serial.println(ylimaaraisia);
}

void Create_class(String typeString, int amount, int courseLength, int mPill, int ePill, int day)
{
    Nimi = typeString;
    maara = amount;
    maara_paiva = courseLength;
    aamu = mPill;
    ilta = ePill;
}

//The arithmetic function that does the math for function void testprintjaljella()
void update()
{
    maara_jaljella = maara - (Days - paivia)*aamu - (Days - paivia)*ilta;
    paivia_jaljella = maara_paiva - (Days - paivia);
    ylimaaraisia = maara - maara_paiva*aamu - maara_paiva*ilta;
}

//lomkLoop is to know how many times we have looped one box and to let the user know if he/she wants to overwrite data in one box
//If it contains any data, if (lomkLoop == 0), it will tell the user the box is currently empty.
//Initializing each class to have values zero for all integers and string to be empty.
Lokero lokero1("",0,0,0,0,0);
Lokero lokero2("",0,0,0,0,0);
Lokero lokero3("",0,0,0,0,0);
Lokero lokero4("",0,0,0,0,0);
int lok1Loop;
int lok2Loop;
int lok3Loop;
int lok4Loop;

//Initializing values for servo
Servo servo;

int servoPin = 9; // Sätt detta värde till pinnens nummer man kopplar servon till
int stepDirectionPin = 2;
int stepPin = 3;

int current = 4; //FAR I MOTSOLS från 1-4
int armOut = 55;
int armin = 180;

int pos;
int gate = 4;

void setup()
pinMode(gate, OUTPUT);
servo.attach(servoPin);
pinMode(stepDirectionPin, OUTPUT);
pinMode(stepPin, OUTPUT);
digitalWrite(stepDirectionPin, LOW);
digitalWrite(stepPin, LOW);
analogWrite(gate, 0);
servo.write(armIn);
Serial.begin(9600);

delay(1000); //Optimerar Steppern
analogWrite(gate, 255); //Optimerar Steppern
}

//Setting up the time according to what the user said. The system is fragile in the sense that any zero integer will make serial monitor skip it
void loop() {
  while(Serial.available()) {
  }

  while(varE == 0) {
    Serial.println("Enter the current hour (24 if 00.xx)");
    Serial.flush();
    varE++;
  }

  while(varE == 1) {
    Hr = Serial.parseInt();
    if (Hr > 0) {
      varE++;
    }
  }

  while (varE == 2) {
    Serial.println("Enter the current minute (60 if xx.00)");
    Serial.flush();
    varE++;
  }

  while (varE == 3) {
    Min = Serial.parseInt();
    if (Min > 0) {
      varE++;
    }
  }

  //This choice will lead to 2 different children nodes if viewed as hierarchy tree and from there you can choose further information
  while (varE == 4) {
    Serial.println("Press 1 to view the medicines inside");
    Serial.println("Press 2 to insert a new medicine");
    Serial.flush();
    varE++;
  }

  while(varE == 5 && varH == 0) {
    varH = Serial.parseInt();
  }

  int Answer;
  int varA;
  String typeString = "";
  int amount = 0;
  int courseLength = 0;
```cpp
int perDay = 0;
int mPill;
int ePill;
int lok;

while (Serial.available()) {
}

// View current box information according to which number between 1-4 the user pressed
while (varH == 1 && varI == 0) {
  Serial.println("Please select which slot to view (1-4), 5 or higher exits the menu");
  Serial.flush();
  varI++;
}

while (varH == 1 && varI == 1) {
  viewslot = Serial.parseInt();
  if (viewslot == 1){
    lokero1.update();
    lokero1.testprint();
    lokero1.testprintjaljella();
    viewslot = 0;
  }
  if (viewslot == 2){
    lokero2.update();
    lokero2.testprint();
    lokero2.testprintjaljella();
    viewslot = 0;
  }
  if (viewslot == 3){
    lokero3.update();
    lokero3.testprint();
    lokero3.testprintjaljella();
    viewslot = 0;
  }
  if (viewslot == 4){
    lokero4.update();
    lokero4.testprint();
    lokero4.testprintjaljella();
    viewslot = 0;
  }
  if (viewslot >= 5) {
    varH = 0;
    varI = 0;
    varE = 4;
  }
}

// Here is the other child node, since the user earlier pressed two we skipped the last part about viewing current box informations
// This child node allows the user to input new information about the pills e.g. when adding a new or removing an old medicine
while (varH == 2) {
  varA = 1;
  while (varA == 1) {
    Serial.println("Please enter the number of the box to which the following specifications apply: ");
    Serial.flush();
    varA++;
  }
  while(varA == 2){
    lok = Serial.parseInt();
    if(llok == 1 || lok == 2 || lok == 3 || lok == 4){
      if (lok == 1) {
      ```
currentlokero = switchContainer(1, currentlokero);
}
if (lok == 2) {
    currentlokero = switchContainer(2, currentlokero);
}
if (lok == 3) {
    currentlokero = switchContainer(3, currentlokero);
}
if (lok == 4) {
    currentlokero = switchContainer(4, currentlokero);
}

Serial.println(lok);
varA++;
}

while (varA == 3) {
    Serial.println("Please enter type of medication:");
    Serial.flush();
    varA++;
}

while (varA == 4) {
    typeString = Serial.readString();
    if (typeString.length() > 0) {
        Serial.println(typeString);
        varA++;
    }
}

while (varA == 5) {
    Serial.println("Please enter amount of pills:");
    Serial.flush();
    varA++;
}

while (varA == 6) {
    amount = Serial.parseInt();
    if (amount > 0) {
        Serial.println(amount);
        varA++;
    }
}

while (varA == 7) {
    Serial.println("Please enter the length of the treatment (in days): ");
    Serial.flush();
    varA++;
}

while (varA == 8) {
    courseLength = Serial.parseInt();
    if (courseLength > 0) {
        Serial.println(courseLength);
        varA++;
    }
}

while (varA == 9) {
    Serial.println("Please enter the amount of morning pills");
    Serial.flush();
}
while (varA == 10) {
    mPill = Serial.parseInt();
    if (mPill > 0) {
        Serial.println(mPill);
        varA++;
    }
}

while (varA == 11) {
    Serial.println("Please enter the amount of evening pills");
    Serial.flush();
    varA++;
}

while (varA == 12) {
    ePill = Serial.parseInt();
    if (ePill > 0) {
        Serial.println(ePill);
        varA++;
    }
}

if (varA == 13) {
    Serial.print("Type of medication is: ");
    Serial.println(typeString);
    Serial.print("Inserted amount is: ");
    Serial.println(amount);
    Serial.print("Length of course: ");
    Serial.println(courseLength);
    Serial.print("Morning Pills: ");
    Serial.println(mPill);
    Serial.print("Evening Pills: ");
    Serial.println(ePill);
    varA++;
}

if ((amount - (courseLength*mPill) - (courseLength*ePill)) < 0) {
    Serial.println("The given amount is smaller than the needed for the course.");
    varA = 0;
} else {

while (varA == 14) {
    Serial.println("Do you wish to overwrite data in one of the boxes, if so answer 1, if not, answer 2");
    Serial.println("Current content: ");
    Serial.flush();
    if (lok == 1) {
        if (lok1Loop == 0) {
            Serial.println("Box is empty
");
        } else {
            loker01.testprint();
        }
    } else {
        lokero1.testprint();
    }
    if (lok == 2) {
        if (lok2Loop == 0) {
            Serial.println("Box is empty
");
        }
    }
}
else{
    loker2.testprint();
}
}
if (lok == 3) {
    if (lok3Loop == 0) {
        Serial.println("Box is empty\n");
    } else{
        loker3.testprint();
    }
}
if (lok == 4) {
    if (lok4Loop == 0) {
        Serial.println("Box is empty\n");
    } else{
        loker4.testprint();
    }
} varA++;

// Reads the input from serial monitor and if the user pressed 2 it will put varA to 0 and therefore skip the whole part about overwriting data
while(varA == 15) {
    Answer = Serial.parseInt();
    if (Answer == 1) {
        varA++;
    } if (Answer == 2) {
        varA = 0;
    }
}
while(varA == 16) {
    if (lok == 1) {
        loker1.Create_class(typeString, amount, courseLength, mPill, ePill, Days);
        loker1.testprint();
        lok1Loop++;
        varA = 0;
    } if (lok == 2) {
        loker2.Create_class(typeString, amount, courseLength, mPill, ePill, Days);
        loker2.testprint();
        lok2Loop++;
        varA = 0;
    } if (lok == 3) {
        loker3.Create_class(typeString, amount, courseLength, mPill, ePill, Days);
        loker3.testprint();
        lok3Loop++;
        varA = 0;
    } if (lok == 4) {
        loker4.Create_class(typeString, amount, courseLength, mPill, ePill, Days);
        loker4.testprint();
        lok4Loop++;
        varH = 0;
varA = 0;
}
}
varE = 4;
varH = 0;
varA = 0;
}
}

//counts the time the system has been on, millis(); function and does the necessary conversion to know when an second, 
//minute, hour and ultimately a day has passed
//this part also manages the output of pills at the right moment
milliS = millis();
Secs = milliS / 1000;
if ((Secs % 60) == 0 && varF == 0) {
    Min++;
    varF++;
}
if ((Secs % 60) == 1) {
    varF = 0;
}
if (Min >= 60) {
    Hr++;
    Min = 0;
}
if (Hr == 24) {
    Hr = 0;
    Days++;
}
while (Hr == 8 && varG == 0) {
    currentlokero = switchContainer(1, currentlokero);
    grabPill(lokero1.aamu);
    currentlokero = switchContainer(2, currentlokero);
    grabPill(lokero2.aamu);
    currentlokero = switchContainer(3, currentlokero);
    grabPill(lokero3.aamu);
    currentlokero = switchContainer(4, currentlokero);
    grabPill(lokero4.aamu);
    varG++;
}
while (Hr == 9 && varG == 0) {
    varG = 0;
}
while (Hr == 20 && varG == 0) {
    currentlokero = switchContainer(1, currentlokero);
    grabPill(lokero1.ila);
    currentlokero = switchContainer(2, currentlokero);
    grabPill(lokero2.ila);
    currentlokero = switchContainer(3, currentlokero);
    grabPill(lokero3.ila);
    currentlokero = switchContainer(4, currentlokero);
    grabPill(lokero4.ila);
    varG++;
}
while (Hr == 21 && varG == 0) {
    varG = 1;
}

/a function for the mechanical outputing of pills, the argument determines the amount of pills of the current type to
void grabPill(int amount) //amount: The amount of pills that should be taken
{
    for(int i = 0; i < amount; i++) { //Repeat amount times
        servo.write(armOut); //The servo moves it arm out
        delay(2000); //Waits for a pill to fall into the "hand"
        servo.write(armIn); //The arm moves back in so that the pill falls down.
        delay(1000); //The servo needs a small break
    }
}

int switchContainer(int number, int currentCont) //number: the number of the container to be switched to, currentCont: current container
{
    int container[] = {0, 400, 800, 1200}; //An array for the steps needed to be taken for the corresponding containers
    int stepCount = 0; //A variable for counting the steps taken
    int target = container[number - 1] - container[currentCont - 1]; //calculates the number of steps needed to be taken to reach the target
    int microDelay = 1100; //The delay between each step; determines the angular velocity of the stepper

    if(target < 0){
        digitalWrite(stepDirectionPin,LOW);
    }
    if(target > 0){
        digitalWrite(stepDirectionPin,HIGH);
    }
    target = abs(target);

    for(;target > stepCount; stepCount++){
        digitalWrite(stepPin, HIGH);
        delayMicroseconds(microDelay);
        digitalWrite(stepPin, LOW);
        delayMicroseconds(microDelay);
    }

    //Returns the current container so that it can be stored in the master code
    currentCont = number;
    delay(800);
    return currentCont;
}