UPPSALA
UNIVERSITET

ISRN UTH-INGUTB-EX-E-2020/001-SE
Examensarbete 15 hp
Juni 2020

Automated test rig for electronic products

August Kollberg
Leonardo Alvarez Urrutia

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Automated test rig for electronic products

August Kollberg, Leonardo Alvarez Urrutia

Testing to ensure the function of PCB:s is a key process that is mandatory for factory constructed boards. Given a large number of test object shows that manual processing is inefficient. By automating the process companies will have the possibility to do this process more efficient, but how can the process be automated for a flexible flow and different PCB designs?

This report describes the work done when designing a prototype unit that is part of an automated PCB-test line. The prototype has the task to sort PCB-cards, approved or non-approved, after the test has been completed.

The system consist of an elevated slider with a trolley whose position is controlled by a stepper motor and timing belt. On the trolley a construction with a linear actuator and robot gripper is placed. The trolley moves to a position were the PCB is picked up, the signal from a terminal informs the system whether the test is ok or not. The card is then placed in a rack. The status of the rack is controlled by a distance measurement, which gives the distance to nearest empty slot. Two Arduinos are used as controllers.

The system was tested in two parts, first each component and program by itself and later the assemble. The result was a prototype that worked but had room for improvement with some parts in need of replacement. Future work would include error signal management, stabilizing the construction and increasing the flexibility.

Handledare: Daniel Carlsson
Ämnesgranskare: Kjell Staffas
Examinator: Johan Abrahamsson
ISRN UTH-INGUTB-EX-E-2020/001-SE
Tryckt av: Uppsala

Förord

Innan ni tar er tiden att läsa igenom denna rapport vill vi passa på att tacka alla som varit med och gett oss uppmärksamhet under vårt arbete. Trots omständigheterna med Covid-19 har vi fått den vägledning och svar på dem frågor vi haft. Vi tackar Daniel, Håkan och Terje på Syntronic AB i Gävle för ni trott på oss och givit oss möjligheten att göra arbetet. Vi vill även tacka övriga kollegor på Syntronic för ni tog emot oss och gav oss de verktyg vi behövde för genomförandet av arbetet. Vi vill tacka vår ämnesgranskare Kjell Staffas som erbjudit laborationsplats och funnits som vägledare under rapportskrivningen.

Detta arbete har pågått under en väldigt speciell tid i samhället, vi hoppas ni och era nära mår bra och ni tar hand om varandra.

Leonardo & August

Vocabulary

I^2C - Inter-integrated circuit, a 2-wire communication protocol

PCB - Printed circuit board

BON - Bed of nails, test fixture for electronic products

ICT - In circuit testing, PCB test that checks shorts, opens and other electric parameters

PWM - Pulse width modulation, method for reducing active power consumption of an electrical input.

DC - Direct current

AC - Alternating current

TOF - Time of flight, method that uses time difference to determine distance

V - Volt, SI unit for electric potential

A - Ampere, SI unit for electric current

Ω - Ohm, SI unit for electric resistance

FSR - Force sensing resistance, resistance that change its value when force is applied on it.

SDA - Serial data line, data line used by I^2C

SCL - Serial clock line, clock line used by I^2C

Populärvetenskaplig sammanfattning

Elektronik är något som bara ska fungera enligt många. Fungerar det inte skapar det ilska och i vissa fall kan ej fungerande elektronik leda till olyckor. Därför är det viktigt att säkerhetställa att all form av elektronik fungerar som det ska genom att utföra test på dem.
Kretskort är något som nästan all form av elektronik är uppbyggt kring. Syntronic AB är ett konsultföretag som bland annat jobbar med test av elektriska produkter och därmed även testar kretskort.

I dagsläget är test av kretskort på Syntronic manuella. Testfixturen som används är en BON-fixtur som bygger på att nålar placeras ner på olika noder på kretskorten och skickar signaler. Dessa signaler testar olika funktioner hos kretskortet och att den utgående signalen ser ut som den ska. Genom automatisering av dessa test ser man till att personal undviker att bli bundna till en teststation och kan utföra andra arbetsuppgifter under tiden.

Testprocessen består av följande steg:

- Placering av kretskort i fixtur
- Stänga/öppna lock för BON-fixtur
- Plocka upp och sortera godkända och icke godkända kort

Denna rapport beskriver arbetet som genomförts av två studenter vid Uppsala universitet vid framtagning av en prototyp som sorterar godkända och icke godkända kretskort. Prototypen bestod av en plockarm som kunde röra sig horisontellt via en slider som agerade som åkdon och vertikalt med en linjär stegmotor. Åkdonet var upphöjt en bit över marken. Systemet styrdes av två Arduinos; En som styrde den horisontella rörelsen och en som styrde den vertikala rörelse och plockarmen.

Kretskorten hämtades upp i ena änden av slidern och lämnas i ett av två ställ; ett för godkänt test och ett för icke-godkänt test. Avlämningsposition i respektive ställ fås via en distansmätning som utförs av en lasersensor.

Prototypen uppfyllde sin funktion men lämnade rum för utveckling. Slutsatsen av arbetet var att idén var bra men mer professionella komponenter

och en stadigare konstruktion krävs för att prototypen ska kunna användas i industrin

Contents

1 Introduction

This report discusses an investigation that looks into the possibilities to automate a test line to increase work efficiency. The investigation is done for Syntronic AB in Gävle Sweden. The test line is used to test electronic products, firstly printed circuit boards (PCBs), and handles varying flow. This work treats the automation of the sorting of approved and non-approved products.

Automation is an interesting subject that more and more companies look in to. When reading about electronics, the subject is up-to-date. With digitization of the society follows a higher demand of electronic products and higher demands that the electronical products work as expected.

For this reason, this project is interesting and not just for the company affected but for society. An automated test line is often very complex and often seen for static flow or bigger volumes.

1.1 Background

Syntronic AB has a test structure that is manual and the line has room for development. Many of those working at Syntronic in Gävle are engineers which means that they have more duties than testing PCB.

An automated test rig offers an opportunity for companies to expand, increase the flow and reduce the amount of monotonously work which may lead to a more attractive work environment.

The technical development in our society is increasing and many parts of the society is going digitized in the future. Nowadays we found electronics everywhere. Therefore, the demand for testing new electronics increases.

1.2 Purpose and problem

The purpose of this project is to investigate the possibilities to automate a part of the test rig. An automated test rig will reduce the monotonously work and the goal is to increase the efficiency of the flow.

Can a test line with various flow be automatized in a way that increases the efficiency while it eliminates the monotonously work?

1.3 Responsibilities defined

This project is a collaboration between two electrical engineer students. Since it is a collaboration, parts of the work are done by one single part. These parts are:

- August: 2.5, 3.2.4, 3.2.7, 3.2.8, 3.2.11, 3.3.1, 3.3.6, 3.3.7, 5.5.1
- Leonardo: 2.3, 2.4, 3.2.3, 3.2.9, 3.2.10, 3.3.2, 3.3.3, 3.3.4, 3.3.5, 5.5.2

2 Theory

This chapter explains components and concepts that the project is based upon.

2.1 Test line

The test line that is used for testing PCB cards and make sure that they fulfil the demands looks as following:

- PCB card is placed in test fixture.
- The test fixture is pressed down ensuring test-nails hit the test object.
- When the test is done, the card is placed in a container for approved/non-approved components

2.1.1 The test fixture

The PCB test is a in-circuit testing (ICT), which is done by applying voltage to different nodes of a PCB with probes to verify function of the card. This can be done in different ways, in Syntronic the testing is done with a bed of nails (BON)[1, pp. 62-76]

A BON is a standard ICT fixture and consists of several probes that are spring loaded. The design and placement of the probe depends on the PCB that will be tested. The BON is pressed against the PCB to ensure that all of the probes have contact.

Depending on the complexity of the test-object the time of the test varies. The tests at Syntronic takes 1-5 minutes depending on the complexity of the test-object according to Daniel Carlsson, employee at Syntronic. It rarely takes under one minute.

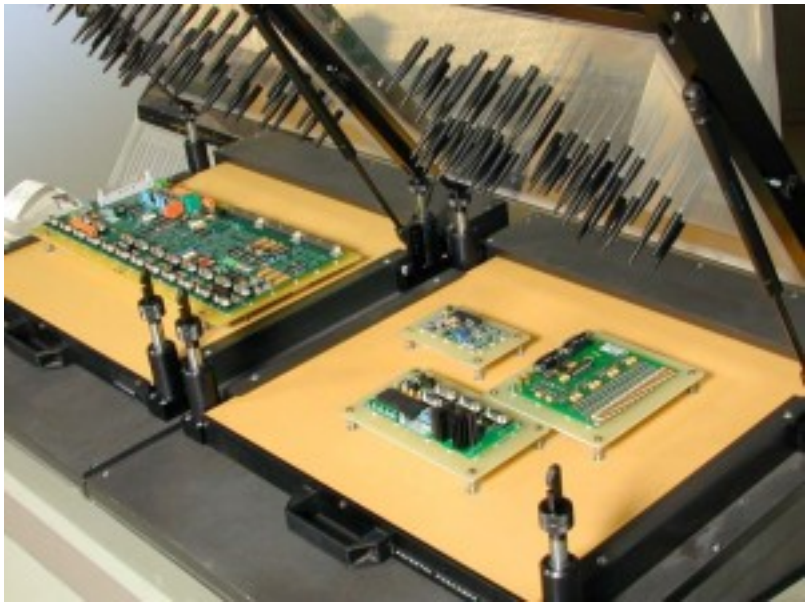


Figure 1: A BON fixture[2]

2.1.2 Sorting

When a PCB has been tested in the BON, the operator directly gets the result of the test. The card is then manually placed in the right sorting pile, approved or non-approved.

2.2 Actuators

In electronics, the DC motor is widely used as actuators, from big robotic structures down to small fans. It uses direct current and converts it into mechanical energy. There are several types of DC motors, each good for different kind of applications.

2.2.1 Pulse width modulation

Pulse width modulation (PWM) is a type of control method for running DC-motors. A PWM-signal contain pulses which width can be modulated. The pulse has two states, high or low. The pulse has a period and a duty cycle. The duty-cycle is how long time of the signal period the pulse is high. A PWM-signal can simulate a lower voltage than what the voltage of the source is[3, pp. 449]. For example if a voltage source is 10V and the PWM has a duty cycle of 50% the effect of the signal will be the same as with a voltage source of 5V.

2.2.2 H-bridge

To be able to regulate a DC motor a driver must be used. This driver must be able to regulate speed and direction of the motor. The H-bridge is often used for controlling motors. It consists of 4 transistors in a H formation with an DC-motor in between. Figure 2 shows how this works.

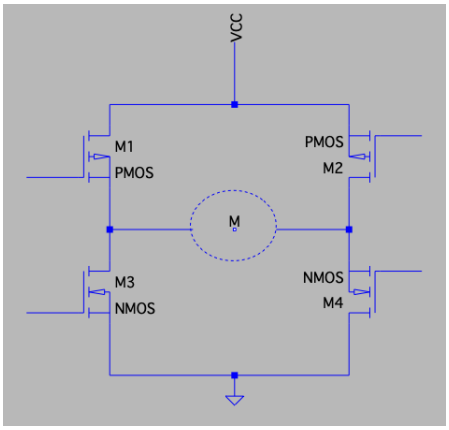


Figure 2: H-bridge

The H-bridge can either be bipolar or unipolar. A bipolar H-bridge means that the upper two transistors in figure 2 has one kind of polarity and the lower two has the opposite polarity. Looking at figure 2, the bipolar H-bridge would have upper transistors of P-type and lower ones would be N-type. The current flow through the motor is controlled by signals on the transistor base. Either M1 and M4 are conducting with M2 and M3 chocked or vice versa[4, pp. 82-83].

VCC is always constant and to regulate the speed of the motor a PWM-signal is often used on the gates/base of the transistor.

2.2.3 Brushed DC motor

These motors consist of a stator, a rotor with current-flowing conductors and commutators which changes the direction of the current in the rotor winding. The rotation of the motor is required by interplay between the magnetic field created by the stator and the force on the conductors. It can be seen as two stationary magnetic fields which interact and creates a torque on the rotor. [4, p. 5-7]

2.2.4 Stepper motor

The stepper motor does not rotate continuously but instead takes steps to complete a full rotation. These motors have a rotor built of several cogged magnets and concentrated windings on the stator creating several phase shifted coil pair, or electromagnetic pairs [5, pp. 6]. The cogs of the rotor are attracted to the active electromagnetic pairs(pair n_i), whilst slight offset from the next pair (pair n_{i+1})

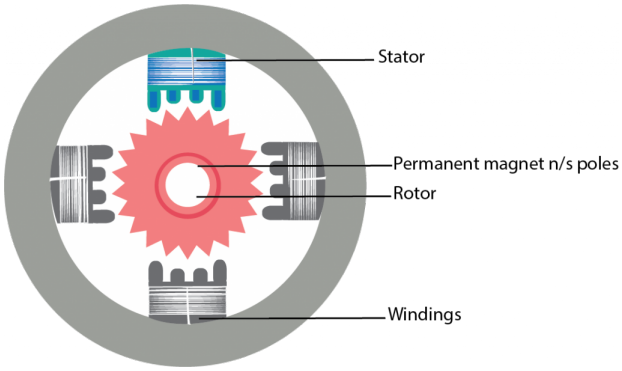


Figure 3: Stepper Motor [6]

Shifting the phase of the electromagnetic field, by turning on the active pair and turning of the next pair (n_{i+1}) the magnetic cogs will attract to the next pair and a rotation is made, is called a step. A full rotation consists of an integer number of steps and in this way the motor can be turned whilst the position is controlled, thereby no sensors are needed giving a open loop.

2.2.5 Servo motor

There are different types of servos, both AC and DC-driven. Some servos provide direct feedback and others do not. The system designed in this project includes a type of servo called hobbyist servo.

A hobbyist servo motor uses a DC-motor, controlled by a PWM-signal. The PWM signal does not, for servo motors, control the rotation speed but instead controls the position of the motor which depends on the duty cycle. A hobbyist servo motor often ranges from 0 to 180 degrees and every degree corresponds to its own pulse width[7, Ch.5].

2.3 Distance measurement

Distance measuring is an important application when working with automation. There is various methods of measuring distance, from manually measuring with rule to digital ways. In automation, it's important to monitor various parameters. For monitoring distance, time of flight(ToF)-sensors are widely used.

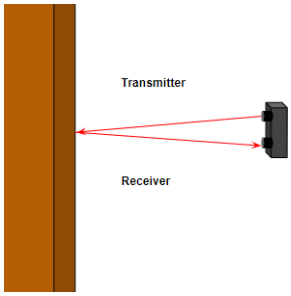


Figure 4: Graphical view of a ToF sensor

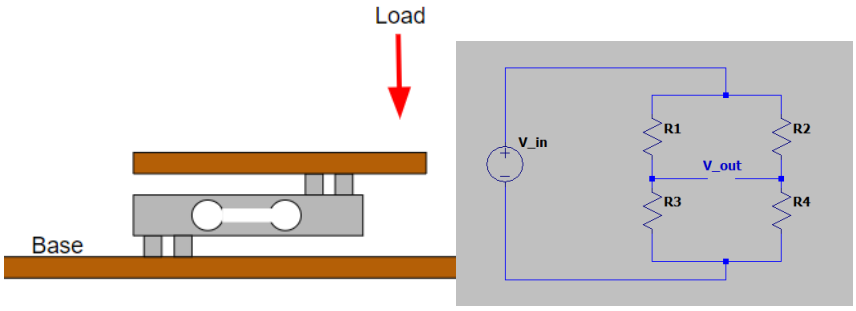
2.3.1 Laser sensor

Laser sensors are a type of ToF-sensors, that with laser pulses calculates distance by measuring time difference between outgoing signal and in going reflection of the signal. Laser pulses are sent towards an object, the laser reflects of the object to a receiver[8]. The distance between the sensor and the object can then be calculated by equation(1).

$$Dist = \frac{v_{light} * time}{2} \tag{1}$$

2.4 Load cell

A load cell is a transducer commonly used in digital weight measuring. They convert force to voltage with help of a Wheatstone bridge, a resistor network. Applied force changes the electrical signal output. The load cell can be combined with an amplifier and micro controller in projects to measure weights. [9]



(a) Load cell (b) Wheatstone bridge

2.5 Force sensing resistance

The Force sensing resistance(FSR) sometimes called piezoresistive sensor is a resistive sensor. The change of resistance is due to the piezoelectrical material it is made of. These kinds of material generate an electrical charge when undergoing stress. One type of stress is when force is applied to the material [10]. When these charges are generated the resistance in the sensor will decrease. The sensor has to be placed in series with a known resistor, with a resistance value much smaller than the max resistance of the FSR. This so you can measure the voltage over the known resistor and see how the voltage variates. In figure (5) is this expressed in a circuit. The voltage can easily be calculated using equation (2).

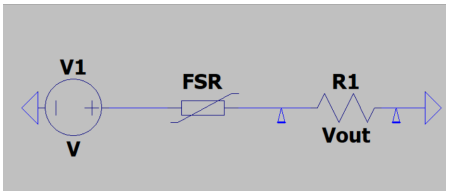


Figure 5: The schematic of a FSR sensor in a circuit

$$V_{out} = V_1 \frac{R_1}{R_1 + R_{FSR}} \tag{2}$$

The FSR is not very accurate and should not be used in a system where accuracy is important. It should not be used in an application with slow changes or static processes.

2.6 Multiplexer

A multiplexer (MUX) is a type of data selector. It is a combinational logic circuit with more than one input but only one output. The MUX switches between different inputs by changing values on the selector inputs, named S_0 and S_1 in figure(6).

The MUX's different addresses indicate which of the input ports that should be represented in the output. The addresses have 'n' bit which represents

the binary numbers. In figure(6) the MUX is an 2^2 ratio MUX which means it has four inputs.[11]

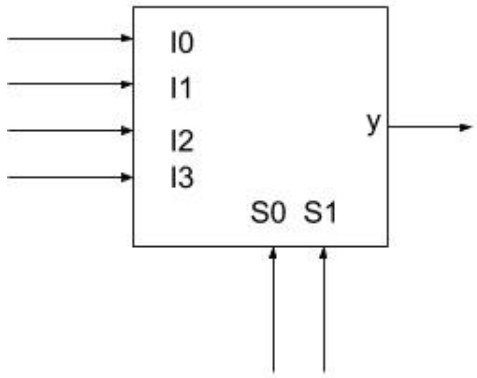


Figure 6: A graphical model of the MUX

2.7 Arduino

Arduino is an open-source electronics platform based on hardware and software often used in small project. The Arduino hardware is based on a micro controller that can read and write signals. The micro controller varies depending on which type of Arduino board that is used.

Arduino manufacture different sort of micro controller boards. In this project, an Arduino Uno and Mega is used. The Arduino Uno uses an ATmega328P micro controller while the Mega is based on ATmega2560 chip. The boards have 14 respective 54 digital ports which can be programmed to be used as either an input or an output. The output from these ports are 5V. 6 respective 15 of these ports can be used as PWM outputs. Both boards have analog inputs, 6 for the Uno and 16 for the Mega, which are used for reading analog signals and writing digital signals.
The Arduino boards uses a 16MHz crystal oscillator. [12]

2.7.1 The Arduino software

The Arduino software is called Arduino IDE and has its own language which is based on C/C++ language. Many libraries used in the C++ language can

be used in the Arduino software

Arduino is open source and many libraries can be found in different communities. These libraries are often developed to be combined with different types of hardware. The libraries simplifies hardware close programming.

2.7.2 I^2C

The Arduino hardware offers connection over I^2C, a 2-wire communication protocol that allows bidirectional communication between master and multiple slaves. For the Arduino, it can be used when communicating with other Arduino devices, sensors and so on. The 2-wire line is possible as every slave has a specific address, the signal from the master consist of a ping and an address. [13]

The two wires used in a I^2C communication are called clock(SCL) and data(SDL). The procedure for when data is sent will begin with the master device sending an attention signal, a start signal, to all slaves. The master then sends the 7-bit address of the device it would like to contact. Along with the address bits the master is also sending a read or write signal bit to the slave device. Each slave device in the system compare the received address with it's own. If they match, the device acknowledges this by returning a signal to the master. The master receives this signal and starts the data transmission. The transmission consists of 8 bytes of data, with every byte consisting out of 8 bits. All the bytes transferred are followed by an acknowledge signal bit. When the transmission is done the master issues a stop signal.

Some devices have a fixed I^2C address given by the manufacturer, when this is the case a multiplexer can be used. In some cases, the predetermined address of an I^2C device can be changed through coding.

The data on the SDA line are valid when the SCL is high. When the SCL is low the change of data is allowed[14, pp.10-11]. This can be seen in figure(7).

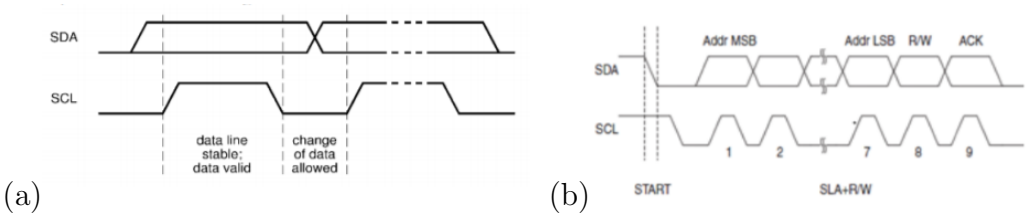


Figure 7: (a)SDA and SCL. (b) The serial communication.[14, P.10-11]

3 Method

This chapter describes the application of the theory and choice of components. Section 3.3 describes the programming done through the project.

3.1 Setup

The system is built around a linear slider. The linear slider has a trolley that moves horizontal with help of a stepper motor and a timing belt. A linear actuator is mounted on the trolley, this actuator controls the vertical motion of the system. On the actuator, a gripper is attached. The system also includes two PCB racks, one for the approved PCB-cards and one for the non-approved. A sketch of the system can be seen in appendix C figure 19.

The system receives a signal, from a terminal, that implies that the BON test is completed. After this signal, the system will get another signal that indicates if the test is approved or non-approved. These signals are triggered by buttons on the terminal seen in figure 8. The terminal is controlled by the Arduino Mega.

Figure 8: The terminal

Depending whether the PCB-card is approved or not approved it is placed in one of two rackets. A laser sensor in the racket where the card will be placed will measure the distance to the nearest placed PCB. From this measurement, the system will know the nearest empty slot and how many steps the trolley needs to slide. An extra check to see if every place in the rack is filled until the closest placed PCB-card is done with a weight measurement given by a load cell. If the weight measurement agrees with the distance measurement, the system will proceed. If not an error signal is sent to an operator. The operator will then have to place the PCB in the right slot and

restart the system. When the check is done the trolley will move to pick up

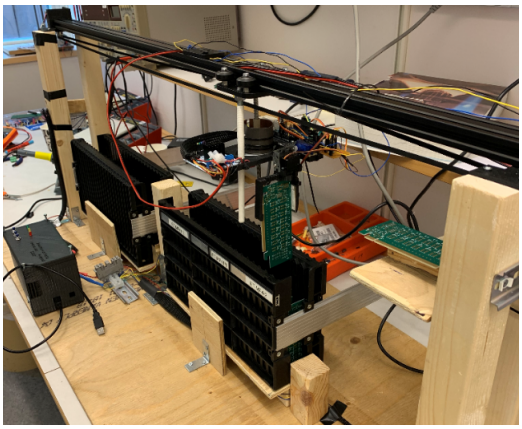


Figure 9: The system in whole

position. The Arduino Mega will control a stepper to step until in pick up position. I^2C will be used to communicate with sensors and the Arduino Uno, which will run as slaves through the process.

The Arduino Uno will be in receive mode until it receives a signal from the Arduino Mega. The Arduino Uno will then control the vertical actuator to given position depending on the received signal. A gripper with two

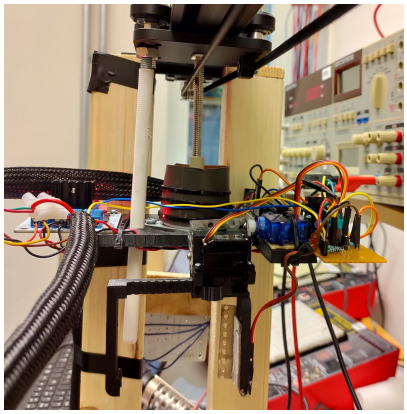


Figure 10: The Trolley

hobbyist servo motors are attached to the linear actuator. One servo motor

will adjust the gripper between two positions. The positions are 0° and 90°. The other servo is fixed to a gear. The gear will move a slider that adjust the width of the gripper. On the gripper, a FSR is attached which works as an indicator for when a PCB-card is gripped. The linear actuator will then move to a new position.

When the pick-up procedure is done the Arduino Mega will receive a signal from the Arduino Uno. Depending on the result of the test and the values from the sensors the trolley will move to the first empty slot in the rack.

When the trolley is in the right horizontal position the Arduino Uno will receive a signal and drop the PCB in right slot. After the PCB is dropped the trolley will return to start position and the procedure will restart. In appendix C figure 20 and figure 21, process scheme for both Arduinos can be seen.

3.1.1 Components in the setup

In figure(11) the system is illustraded. Components and their functions:

1. Camera slider used for horizontal movement. A timing belt attached to a motor moves the trolley along the slider
2. The trolley.
3. Rack where the approved PCB-cards are placed
4. Rack where the non-approved PCB-cards are placed.
5. The terminal, signals the outcome of the BON-test.
6. Pillar where the distance measurements sensors are placed.
7. Pick up place for the PCB-cards.

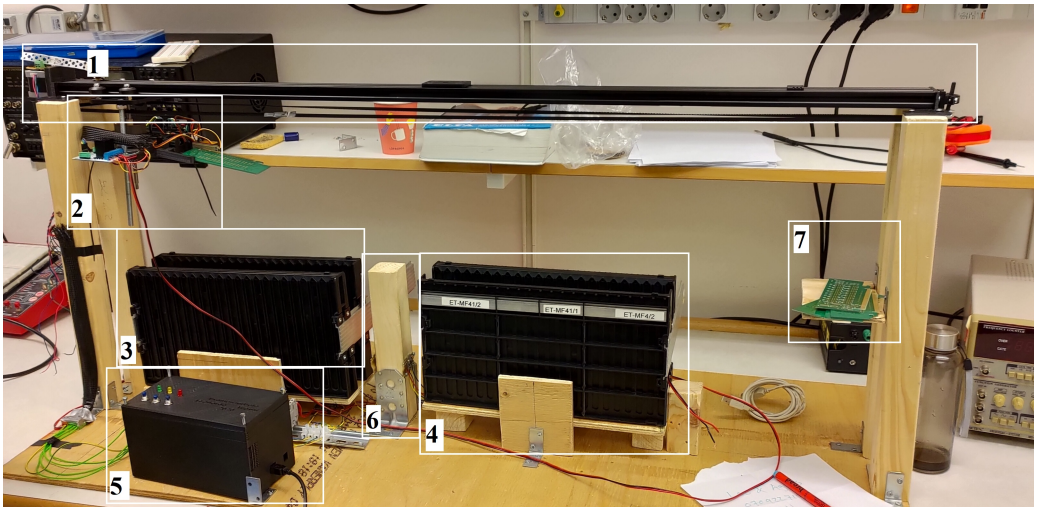


Figure 11: The main components of the system

3.2 Hardware

This chapter presents the hardware used in the project.

3.2.1 Base

The model was built on top of a base made of a MDF-board 1200x350mm. Two wooden studs, 500mm in height, were placed on both short sides of the board used to elevate the slider from the base. The elevation ensured that the slider with gripper and linear actuator were able to move freely over the racks.

The racks were placed centered under the slider along the long side of the base. They were placed on an MDF-board of 350x45mm each. With one of these MDF-boards placed on top of a load cell and the other was elevated to ensure that both boards were at the same height.

3.2.2 Slider

The slider used in this project was a Ratrig camera slider, used for sliding cameras when needing a smooth horizontal movement. On the slider, there is a trolley attached which can move horizontal on the slider. The ready-to-use

slider was chosen to minimize time spent on mechanic work and for its low friction.

3.2.3 Stepper motor

For the horizontal movement a stepper motor, more specific the Nema 17 17h2A4417, was used, which is a two-phased stepper motor. The motor is rated for 1.7 A and takes 200 steps per rotation with a step angle of 1.8°. It provides an accuracy of ±5%. The motor has four wires, two for each coil were the wires corresponding to each coil can be identified by measuring the resistance over the wires. Each phase has, according to the retailer[15], a resistance of 1.25Ω. Measuring resistance between each wire gave:

- Red: A^+
- Blue: A^-
- Black: B^+
- Green: B^-

Where A and B corresponds each coil.

3.2.4 Vertical movement components

The components of the vertical movement are displayed in figure(12).

1. Arduino Uno. The micro controller that controls the vertical movement
2. The Velleman L298N motor driver.
3. Luxorparts Micro ServoSG90. Named *servo one* in this project. The Servo controlling the position of the gripper.
4. Seeed mini-series ES08A servo. Named *servo two* in this project. The servo controlling the width of the gripper.
5. The L7805CV voltage regulator
6. The RS-pro 42DBL10C2U-L linear actuator

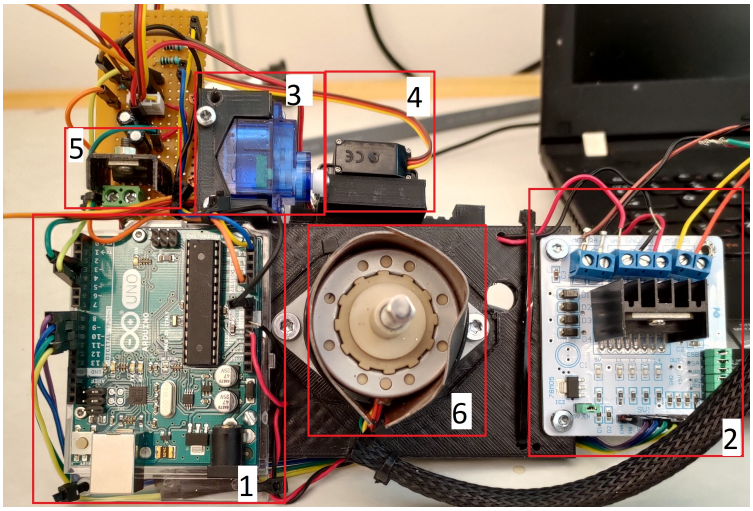


Figure 12: The different parts in vertical axis

3.2.5 Linear actuator

The vertical movement were accomplished by using an electrical linear actuator, for this project the 42DBL10C2U-L by RS-pro. This actuator is a 4-phase unipolar actuator with a lead screw that moves up and down with the rotation of the actuator. It is rated for 12V/10W and has a load limitation of 100N. The 42DBL10C2U-L is as mentioned a unipolar and has 6 cables in to its coils. The datasheet[16] gave:

- Red/Green: $12V$
- Brown: A^+
- Black: A^-
- Yellow: B^+
- Orange: B^-

Just as the stepper motor, this actuator is easily controlled with an Arduino and a motor driver.

3.2.6 Motor driver

The motor driver used in this project for both the vertical and horizontal movement was the L298N board. The L298N is an IC circuit build on the H-bridge principle mentioned under theory 2.2.2. It offers possibility to drive two DC-motors which means that it also works for controlling a Stepper-motor. The L298N can be found as ready-to-use board, in this project Velleman board, which reduce the time spent on soldering a whole circuit to follow the IC.

It is rated for 5-35V and maximum 2A. [17]

3.2.7 Servo motor

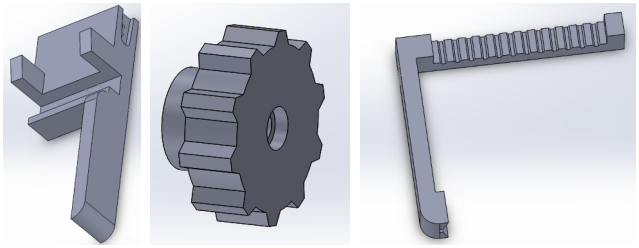
The two servo motors in the gripper are hobbyist servo motors. More specific one *Seeed mini-series ES08A* and one *Luxorparts Micro ServoSG90*. The servos are rated for voltages between 4.8-6.0V. The rotation of both are counter clockwise with the position pulse width between 1.5-1.9ms for the Seed mini and 1-2ms for the ServoSG90.
The advantage of using these servos are their size and holding torque. The Seed mini weights 8.5 gram with a holding torque of $1.5/1.8 kg{\cdot}$cm [18] and the ServoSG90 has a weight of 11g and a holding torque of $1.8/2.4 kg{\cdot}$cm [19].

- Red: Power
- Brown: GND
- Yellow/Orange: Signal

These servo motor are easily programmed with the Arduino servo library.

3.2.8 The gripper

The gripper used is a three-part, 3D printed, assemble designed in Solid-Works. First part was the base on which the Seeed mini ES08A servo motor was placed. On this motor the second part, a gear, was attached. The gear controlled a slider, the third part, that could slide and adjust the width of the gripper. The assemble was attached to an arm that was fixed to servo one. In figure(13a, b, c) the three main parts of the gripper are shown. The arm that connected the gripper with the rest of the system were placed on



(a) Base (b) Gear (c) Slider

the base. In figure (13) the 3D model is assembled without the servo. The gripper grips the card on the sides of the card. With the card sliding in the small slot seen in the slider.

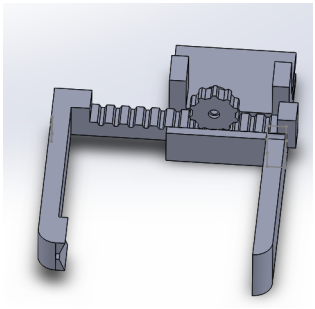


Figure 13: The assemble of the gripper

3.2.9 Distance measurement

Having control on the number of PCB cards placed in each rack is one of the more important factors in this project. These measurements have to be precise and stable and therefore the VL53L0X by Adafruit is used. The VL53L0X is a ToF distance sensor, described in theory 2.3.1. It communicates over I^2C. Two of these sensor were used, one for each rack, and they

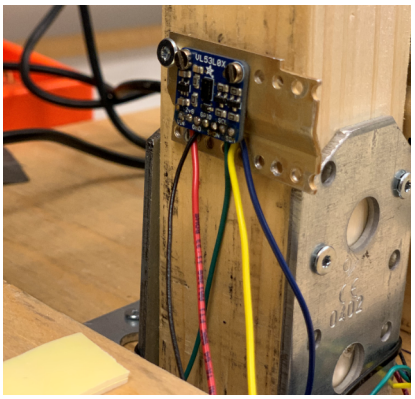


Figure 14: VL53L0X, ToF-sensor, in place

were placed on a small pillar seen in figure(14) one for each rack. The sensors used the same I^2C address which meant that a MUX had to be used to ensure that communication worked as wanted. The signals went through the MUX to the Arduino Mega, where the signal was converted to a distance.

3.2.10 Weight measurement

To ensure that the measured distance to nearest card gave the correct number of empty slots, some sort of control method had to be applied. The choice landed on weight measuring. For this task, the TAL220 was combined with a ready-to-use load cell amplifier board with a HX711 IC. Both were supplied with 5V from the Arduino.

As mentioned in theory part 2.4, the load cell works as a Wheatstone bridge were applied force changes the output voltage of the load cell. The four wires of the load cells were connected to the amplifier and the Arduino was connected to the amplifiers four inputs: VCC, GND, DAT, CLK.

3.2.11 Force sensing resistor

To check if the PCB is gripped a FSR was attached to the griper. The sensor is a FSR02CE and is 600.30mm long and has a width of 15.20mm, the length can be adjusted by cutting the sensor. It can detect weight between 20g and 5kg and has $10M\Omega$ resistance when force less than 20g is applied.[20]

In this system, the FSR02CE is placed in series with a $10k\Omega$ resistor. This means that the voltage over the 10KΩ resistor is almost zero when no force is applied according to equation(2). When 5kg is applied the resistance is really low and the voltage over the 10KΩ resistor will almost reach the value of supplied voltage. The setup can be seen in figure(5)

3.2.12 Multiplexer

The communication protocol I^2C used for communication between Arduino Mega, Arduino Uno and the ToF sensors requires that every component have a unique address. The ToF sensors are of same type and have a fixed I^2C address, therefore an I^2C multiplexer were used to assign each component a unique channel.

The MUX that were used was a TCA9548A I^2C multiplexer breakout board by Adafruit. A board that offers connection with up to eight I^C devices, disregarding if their addresses are identical or if they have different addresses. The I^2C address of this board can be set from 0X70 to 0X77 by three analog inputs were different combinations on the inputs give different addresses, by default 0X70.
The MUX was connected as following:

- VCC - 5V
- GND - Ground
- SDA - Arduino Mega SDA
- SCL - Arduino Mega SCL
- SDA6 - Sensor 1 SDA
- SCL6 - Sensor 1 SCL
- SDA2 - Sensor 2 SDA
- SCL2 - Sensor 2 SCL
- SDA1 - Arduino UNO SDA
- SCL1 - Arduino UNO SCL

3.3 Code

All code was written in the Arduino IDE using the Arduino programming language. The code for the project is based on several smaller codes, one for each part of the project; Horizontal movement, vertical movement, servo motors, distance measurement, weight measurement, communication between Arduinos over I^2C and status of each rack. Since the Arduino board uses a single core processor, the code follows a sequence and can only do one process at the time.

3.3.1 Communication between Arduinos

The communication between the two Arduinos was over I^2C. This was attained with the 'Wire' library. When using the 'Wire' library a master-slave configuration is required. In this case the Arduino Mega was the master. The Arduino Uno was in slave mode together with the ToF sensors. Since a I^2C multiplex is used, only one address is needed, it was set to 0X70.

In the 'Wire' library the master either writes or read data with two different methods. The slave is controlled by the master and can´t write or read by its own command. It has to request its master to either write or read value from the master. The 'Wire' library has methods that can receive events and request event. To use these methods, it is necessary to have another method in the argument.

If the slave wants to receive an event from the master the method in the argument has to have the read command in it. This so the slave can read the data that is sent from the master. If the slave requests an event from the master the argument method needs to have the write command in it. The methods used by the slave can be seen in appendix B figure (b)2.

3.3.2 Multiplexer

The code for the multiplexer is written using the 'TCA9548A'-library and chooses from which channel the I^2C data should be read. The code is very short and easy and is as mentioned only used to ensure that the correct data is received by the Arduino Mega when reading it I^2C line.

3.3.3 Distance measurement

The code for the ToF-sensors is written with help of the 'Adafruit_VL53L0X.h'-library which includes methods for converting voltages to distance. Measurements of distance are made once per 'loop' of the process, more specific when the PCB-test is done and the terminal signals the Arduino Mega whether the test is OK or NOT OK.

The code is written as a state machine having three different states: one for idle, one for reading sensor_1(OK test) and one for reading sensor_2(NOT OK). Both states take ten measurements of the sensor readings, for more accurate values, and calculates the mean of these readings. The calculated mean value is then compared to an array, containing fixed values corresponding distances to each slot of the rack, and returns the index of the nearest empty slot. The code can be found in appendix A figure (a)5 and (b)6.

3.3.4 Weight measurement

For the weight measurement, the code is written with a library for the load cell amplifier:'HX711.h'.

The code first initiates the load cell, after that a calibration is required. For this a calibration factor is used to reach the best possible values from the measurements. The calibration factor was generated by a calibration code downloaded online. Last, an offset and start weight was set. These parameters were defined in the setup.

The weight was measured once the test was approved and a distance to the nearest empty slot was measured by the ToF-sensor. The measured weight is then divided by the card weight to see if the number of cards in place agreed with the distance measurements. If it does, the position is saved and sent to the method for horizontal movement.

3.3.5 Horizontal movement

For the horizontal movement, the code is written using the 'Stepper'-library, which includes a method for stepping the motor with one or several steps at the time. The code has a step counter which changes for every step taken, were step zero is the 'start'.

The code also includes an array of fixed values of steps, each corresponding to a position and slot of the rack. The index of the nearest empty slot was used to determine how many steps should be taken in the array called 'horisPos' seen in figure((a)1) in appendix A. The values in this array represents steps needed to be taken to reach each position. The motor then starts to step. When the correct number of steps are taken, a signal is sent to Arduino Uno. The code can be found in appendix A figure (b) 4.

3.3.6 Vertical movement

Just as the code for the horizontal movement, this code uses the 'Stepper'-library. This motion is controlled by the Arduino Uno. The stepper motor has a method where 5 positions are pre-determined. The argument in this method is a stepper and an integer which indicates the final position. A step counter variable is also initiated in the begin of the code. The 5 different position are:

0. The calibration point and where step calculation starts (0 steps). This point is -450 steps from the top of the linear actuator.
1. Middle station (-500 steps).
2. Pick up position (-2100 steps).
3. Position of the gripper after gripping the PCB (-1000 steps).
4. Position of the gripper after dropping the PCB in the racket (-1500 steps).

The amount of steps the motor shall take is the difference between the required position and the step count variable. After moving to a position, the step count variable will be changed to the step value corresponding to the position.

The stepper will move to the given position when the Arduino Uno receives a signal from the Arduino Mega. The signal received is a byte indicating how far in the process the system has come. The code can be found in appendix B figure (a) 7.

3.3.7 Gripper

The gripper consists of one Seeed-mini series ES08A servos and one Luxorparts Micro ServoSG90. These are easily controlled with the help of the servo library. In figure (12) the gripper is shown from above. Servo one is controlling the rotation of the gripper. The start angle of this gripper was zero degrees. When a signal is received from the Arduino Mega indicating that slider is in position this servo will turn to 90 degrees, so the gripper can grab the PCB.

A signal is sent to the Arduino Mega once the gripper has rotated 90 degrees, which will make the stepper motor controlling the horizontal movement to step. When the gripper is in right horizontal position a signal will be received from the Arduino Mega and the gripping process starts. Servo two will step one degree at the time and read the value from the FSR. The Arduino will then calculate the mean value of 10 readings. The mean value will then be added in to an array with five spots.

A method in the code will then return the minimum and maximum value of the array. If the minimum value of the array is greater than 400 and the difference between maximum and minimum value is less than 10 the method will return true which indicates that the PCB is gripped. If this method returns false the loop will restart with servo 2 taking another one degree step.

When the PCB is gripped and the vertical position is right Servo one will go back to a zero-degree angle one step at the time. Decreasing the risk of dropping the PCB due to rotation of the gripper.

When the trolley is in the dropping position the Arduino Uno will receive a signal from the Arduino Mega. When this signal is received Servo two will go back to zero degrees and the PCB will be dropped into the right slot of the right rack. The code can be found in appendix B figures: (b)2, (b)4 and (a)5.

3.4 Test

The automation is supposed to improve speed and efficiency of the line, therefore process-time for the automated process was recorded. For the au-

tomated test, time was recorded from the second that the signal "test done" registers until the PCB-card were placed in the right place on the rack. This was done 12 times and mean value of the iteration time were calculated

The test was performed in three different speeds on the horizontal movement and three different speeds for the vertical movement. This to see how the iteration time and accuracy of the system would be affected by these factors. To measure the accuracy the following parameters were observed.

- Number of misplaced PCB:s
- Number of dropped PCB:s
- Number of correctly placed PCB:s

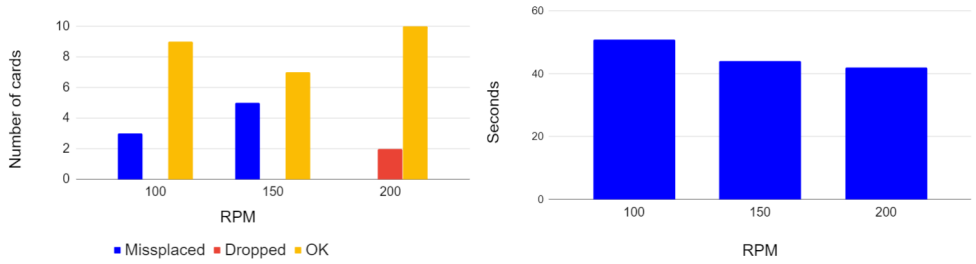
4 Results

The finished system was able to pick up a PCB-card at a pick up-place and drop it in a rack for approved or non-approved cards given a signal from the terminal. The construction of the system made it impossible to fill both racks. The communication between Arduinos worked without problem. Both the vertical and horizontal movement worked as planned and we found no need to have an extra feedback for the movement.

At the end, no weight measurements were made which meant that the status of the racks were controlled by the distance to nearest empty spot, given by the distance measurement. The function of the gripper worked as wanted, even though the pressure measurements weren't used as planned. The distance measurement worked almost perfect, one slot were skipped due to measurement issues.

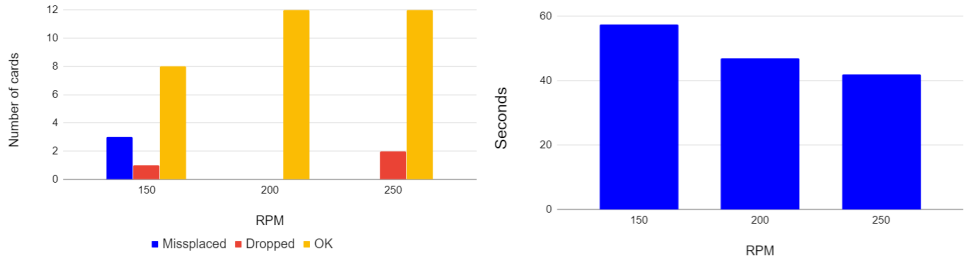
4.1 Test results

Tables below present the result of the test mentioned in section 3.4.



(a) Outcome of test (b) Process-time

Figure 15: Test result for constant vertical RPM and varying horizontal RPM



(a) Outcome of test (b) Process-time

Figure 16: Test result for constant horizontal RPM and varying vertical RPM

5 Discussion

The goal of the project was to develop an idea on how to automate the sorting procedure in Syntronics PCB-testing and to build a prototype. Automated process would lead to a more time-effective process where the need of constant presence of an operator is not needed.

5.1 Test results

The tests were performed to see how the process would be affected by changing the speed of the movements. Both the accuracy and the process-time differed by changing the speed, were the last mentioned was expected before performing the tests.

The time difference was most affected by the vertical movement. The biggest impact on the accuracy of the system was seen when the horizontal speed was decreased and the vibrations of the stepper motor increased. The increased vibrations lead to a noisier system and also shook the whole base. The vibration of the base affected the placement of the racks and the PCB-placement in pick-up position which meant that the cards were gripped in different ways for each iteration.

In the test with 200/250 RPM two PCB-cards were dropped, this was due to human error. The error made concerned the "pick-up" slot, which hadn't been attached correctly to the rest of the structure, this lead to cards not being picked up at all and was registered as a 'drop'

5.2 Time efficiency

The system as whole had a process-time that were slower than the manual, with one big flaw. Which could be identified during the test.

The use of an electrical linear actuator led to a slow but precise vertical movement which was acknowledged early in the project. The actuator used, 42DBL10C2U-L, was a component that was found on site and first and foremost was used to represent the idea. A substitute would be to use a faster actuator that handle less force since the process doesn't require too much force. On the other hand, a faster actuator demands a higher input power

which also means that the motor driver would have to be replaced. A more powerful driver would be a necessary substitute for future use.

On the other hand, the BON-test takes longer time than the sorting-process, which means that a slow sorting-process would not affect the whole test-line. The tests at Syntronic takes 1-5 minutes and the process for the system takes 50 ± 10 seconds depending on the motor speed. Due to this the time efficiency of a full automated test-line may not be affected much by the sorting-process. When regarding a full automation of the test-line, the prototype discussed in this report may be useful.

5.3 Reliability

There were some flaws which led to varying results, where most of the flaws easily could be corrected when considering a long-term solution. The material in the base of prototype was one of the parameters that affected the reliability most. Since the pillars, on which the slider was elevated, weren't hundred percent stable the placement of the gripper could vary. A construction made of aluminum would offer more stability.

Another factor that was affected by the choice of material was the weight measurement. The load cell was attached to MDF-boards as seen in figure (a) under section 2.4, the placement affected the readings negatively. Placing a card on one side of the rack gave different reading than if a card were placed in the middle of the rack. Applying force also lead to small deformations on the MDF-board which the load cell interpreted as an extra load. Therefore, these measurements were ignored. For future use, the setup of the weight measurement should be changed. Alternatively, the status of each rack slot could be controlled using pressure sensors.

The use of stepper motors led to a very precise movement on both vertical and horizontal axises which was key to reach each position. On the horizontal axis, the Nema 17 was an excellent choice offering both speed and precise movement, the joining of the timing belt can be improved.

5.4 Flexibility

The system had several components that were 3D printed, which could be switched out depending on the size of the PCB-card that is tested, one example of this is the gripper that were used. A gripper is easy to design and can be 3D printed on site. The racks that were used can also be adjusted in width, if the test object is wider the rack is adjusted to fit. One problem is the construction of the base, each rack is placed on an elevated MDF board which is fixed in size so a wider rack would need a larger MDF board.
Both actuators used in the project can handle more force so the movement would not be affected with a larger card.

For full flexibility, more advanced programming is needed. Since the distance to each slot, and steps required to reach this slot, were fixed in arrays the values of these arrays would need to be changed in case that another rack with more distance between slots would be used. One option is having different arrays for each rack. Another option would be to look at the possibilities to 'learn' the Arduino the distance between each slot by differentiating the distance at 'start' and the distance when the first card is placed, however this option would need more advanced programming.

5.5 Gripper

The gripper used in the project, described in method 3.2.8, was a gripper inspired by solutions found on diverse websites. One problem was that the gripper was limited in width for this PCB or others with similar size. The PCB sizes can vary which means that this gripper doesn't offer full flexibility. If Syntronic would like to proceed with this idea the gripper has to be a more flexible. It could be 3D printed, or an industrial made gripper. Construction of the gripper is not the most important thing, the most important aspect is its flexibility and robustness. The group members had different thoughts about the gripper, each thought discussed below.

5.5.1 August thoughts

To makes this process economical- and time sustainable you either have to have a flexible gripper or an easy modified gripper. To buy a new gripper or to 3D print a new gripper for every different kind of PCB would not be

considered sustainable in aspect of economy or time.

To have personnel working on making a gripper would be both expensive and waste of time. If this is the case, the test could equally be manual and you would save money and time. One solutions could be to have three kinds of 3D-printed grippers for different sizes; small, medium and large.

Another kind of gripper that could be used is a gripper that has four gripping points and is bigger. It can expand and grip a wider range of PCB-cards. Another solution would be to grip from above with some kind of vacuum gripper. A problem that can occur while using this sort of gripper is that it may damage the soldering on the board. One way of making it work is to program the gripper to have pressure points where there's no components.

5.5.2 Leonardos thoughts

The gripper is designed in such way that it easily could be modified. Most PCB cards can be gripped from the sides just as the cards tested in this project, the limitation of flexibility lands on the range of width. The gripper is an assemble of three parts, where the slider is the only part affecting the grippers width. One could use the same base and gear but change the slider to match the width of the test-object. One parameter that may affect the change of the slider is the holding torque of the servo motor, for bigger PCB cards this parameter should be regarded.

Even if the gripper is flexible, some parameters in the code would need to be changed.

5.5.3 Modified program

The first program that was coded and described in section 3.3.7 was not so reliably. Sometimes the pressure on the FSR was high enough to return true even if the PCB was not gripped good enough. This was due to the card not being positioned in the small slot seen in figure(13c). This lead to the PCB-card being dropped from the gripper when it was lifted from pick up.

The program for the gripper was therefore modified so that the gripper would rotate one degree at the time up to 180 degrees. The FSR then made its mea-

surement and if it felt a pressure it would return true. If the pressure was zero it would return false.
The data of implying if a card was gripped or not was not processed but in future this would send a fault signal if FSR code returned false. The gripper was more reliable after these adjustments.

5.6 Programming and micro processor

The Arduino boards used in this project had both pros and cons, they had limitations but also simplified some parts. Looking at this project as a simplified prototype, the Arduino boards are sufficient. They are simple to program, help can be found online and more sensors or shields can be added. For example, a scanner could be added if one would like to scan a QR-code on the PCB for each test.

Since the Arduino boards use a single core processor, they are unable to perform more than one task at the time. This lead to a system that were slower and were unable to keep track if an error occurred while a process was running. So, if a rack were picked of position after distance readings were done, the system would still believe that the rack was in place and would run as normal. This would cause big problems if the system were to use in industry, this is something that would have to be addressed in future work.

The code is a cluster of several smaller codes which most are written using libraries found online. These libraries often priorities simplicity over speed and the result is a slower code, but for a prototype the code is sufficient and it is simple enough for anyone to use and modify. The code may be slower with these libraries but the system itself is critical in seconds not microsecond or milliseconds. Therefore, the libraries won't affect the process-time of the system much.

5.7 Manual vs Automated

This project is a part of a bigger project in automating a whole process and for this reason it is hard to compare manual work to the automated work. Putting a PCB-card in a rack is process that does not take long time nor demands a single personnel to perform, the process can be done while having

other tasks. But when looking at the whole test-line being automated, the sorting process is a must to include to ensure a full automation.

5.8 I^2C

The communication protocol used in this project, I^2C, was shown to be a good choice. It had good response time and was reliable. Some minor issues were encountered. While communicating between the Mega (Arduino) and Uno, the servo motors started to twitch. This was because of the clock signal of the I^2C signal interfering with the pulse signal controlling the servos, this was solved by adapting the code and changing the signal pins from the Arduino. The issue was based on that the chosen PWM outputs to the servo motors had clock pulses that were influenced by the I^2C communication. To avoid this the PWM pins were changed to digital 9 and 10 and used a library that disabled all 'behind the scene'-interruptions made by the I^2C communication.

5.9 Setbacks

As in all projects, setbacks occurred that prolonged the time spent on the project. These setbacks were not only negative, they were also a good lesson.

5.9.1 Motor driver

The motor driver used in the project was based on the L298N driver, rated for 8-35V and 2A which were supposed to be enough for both actuators used, still problems occurred twice with the driver. The first time might have been a simple mistake were attention was lost and the current peak was too high for the driver to handle, it burnt. The second setback is still a bit of a mystery, the driver had worked perfectly for a long time and at the end of a work day it stopped working. Small investigation was made and no concrete answer were found, it might have gotten overheated.

5.9.2 Weight measurements

To ensure that the right number of cards was placed on the rack an additional measurement, beyond distance measurement, were planned. The weight measurement were supposed to be a double check of the rack to ensure that all

the slots up until the nearest empty slot were filled. When testing the load cell by itself it worked just as wanted, the load cell 'learned' the weight of the card by registering change of weight, saving it as a variable 'previous weight' which later was compared to a new weight. Card weight was then used to calculate how many cards were placed on the rack by dividing the total weight with the weight of the card.

When combining this code with the code of the distance measurement each iteration became very slow and that the code had to be re-written. The new re-written code was built on a predefined card weight and calculated the number of cards in rack by dividing total weight with the predefined card weight, it was still slow but worked. At the end, the measurements were ignored, this because of bad construction giving unreliable values and because it slowed down the system to much.

5.9.3 3D-printing errors

Due to lack of experience of 3D printing some errors occurred. The first error occurred because of how the part were printed. They were placed in a way which made it very hard for the 3D printer to print and the model were incorrect. After some failed prints, placements were corrected so that the printer could work as planned.

Another setback when using the 3D-printer was that error margins were not include when designing the parts in SolidWorks. For example, the slider in the gripper which had to be able to slide in to a slot of the base. First print resulted in a slider unable to fit in the slot due to some small error margins. Learnt from this was to have a margin of at least 0.5mm.

5.9.4 Servos not working as they should

The program for the vertical movement worked good when the Arduino Uno was not communicating with the Arduino Mega via I^2C. It moved vertically without problems and the gripper worked as it should. When the I^2C communication was initialized the servos on the gripper started to twitch. Every time the Uno was receiving data from the Mega both servo one and servo two made small twitches which displaced. When looking at an oscilloscope a small interference of the duty cycle of the control pulse to the servo making

it unstable was seen. The issue was due to some disturbance.

After some troubleshooting, this was solved by changing the ports of the servos. The library that helped controlling the servos was also changed. This to a library where the I^2C clock and the control pulse of the servos were not influenced by each other.

5.9.5 Burnt multiplex-circuit

One setback that didn't create a huge issue but that was a bit annoying occurred at the end of the project. Some modifications had been done on the vertical axis and a screw was lost. After looking a while without finding it, a new screw was used.

When later powering up the system, the I^2C communication didn't work. The rack was removed to access the circuit and the lost screw had fallen under the elevated multiplex-circuit and shorted the whole circuit, which meant our multiplexer was burned. Fortunate enough the I^2C address of the ToF sensors could be changed by rewriting the code

6 Future work

The work described in this report is just sub project of a bigger project. To finish the whole test-line more processes would need to be automated, placing and picking PCB-cards from/to the BON-fixture as well as a process for how the pressure from the nails should be applied. A scanner should also be implemented in some stage of the process, offering bigger control over the flow.

All this has to be integrated to the test software itself so it could communicate the result of the test. The controlling hardware used would have to be compatible with the software that is used in the test.

A graphical interface for this is also needed so an operator can supervise the test. This could be some sort of remote control with display so the engineer responsible for test can work with other tasks and still be able to watch the test.

6.1 Construction

As a whole, the construction should be made more robust. This project is only seen as a prototype and therefore many improvements can be done on the 'base' construction. For example, the choice of material should be regarded, aluminum profiles would make it more stable and enable the possibility to use 'sliders' for holding sensors and racks in place. The wire management should also be looked at, using a cable chain on top of the slider is recommended.

The joint of the timing belt needs to be addressed since the reinforced part makes the band stiffer at just that place. The stiffness might affect the step counter of the horizontal stepper motor making the placement vary. For a more long-term solution, the timing belt should be changed or joint in a better way.

6.2 Sensor measurements

The use of a single sensor measurement for this type of process is not enough, given more time the group would look at using pressure sensors to control

how many slots of the rack has been filled alternatively rethink the weight measurements. Implementing optical sensors for control of the rack is a recommendation given by the thesis reader that should be considered in future work.

6.3 Motor driver

Since the quality of the motor driver was shown to be unreliable, looking at a more robust stepper driver would be recommended. For example, a DM556[21] alternatively controlling the enable pins of the L298N to ensure that it doesn't overheat.

6.4 Bluetooth communication between Arduinos

To minimize the amount of wires, making the construction look cleaner, one should consider to use Bluetooth communication between the Arduinos. Using Bluetooth was an idea in the early stages of the project. Easy-to-use Bluetooth component compatible with Arduinos could be implemented in this project. The I^2C channel between the Arduinos would be removed and data would be sent over Bluetooth instead.

6.5 Robot Gripper

To ensure that the solution is flexible the user should be able to switch between several grippers making it possible to grip different kind of PCB-cards. The construction as built right now is developed for a specific kind of card were the gripper is attached to an arm via a servo joint. The force placed on this joint could, with increased card weight, become too high and therefore several points of the gripper should connect to the vertical platform, in this way the possibility to change gripper would be more flexible.

6.6 Error signals

Even in the safest of system errors may occur. Error signals should be implemented so that the errors could be handled if they should occur. Some error that may occur in this system are:

- PCB not gripped
- Dropped PCB
- Slot not filled
- Rack not in place

Each of these errors should also lead to the system going into 'standby'-mode. This would then be signaled to an operator via the graphical interface. The operator would then need to fix the error and reset the system manually.

7 Conclusion

Since automating the whole test-line is a big project, including the work presented in this report, it is hard to make the conclusion if it would be profitable to fully automate the test-line for Syntronic. To fully automate the whole process would lead to less monotone work and furthermore making it able for personnel to work with other tasks, but economical it's hard to determine whether or not it is profitable.

If the test-line at Syntronic would be automated, the idea and work presented in this report could be used but modifications would have to be implemented. First and foremost, the base structure would have to be more stable and offer more flexibility in rack size's and placement of distance sensors. Some components would have to be switched out to more robust components of higher quality.

Giving more stability to the base would result in a more stable gripper, but the gripper and/or the structure moving on the vertical axis would also need to be modified, making it smaller and more flexible. The process-time was long but in a fully automated test-line it would not matter since the PCB-test itself has a longer process time.

The project showed that this process never could be done faster by an easy automation than by a human. If Syntronic would be interested in investing in a more expensive robot arm, it would be possible to decrease the process time.

8 Reference list

References

[1] *Stephen F. Scheiber*, "Building a successful board-test strategy", 2th ed., Elsevier, 2001

[2] *Seica*, "Parametric and in-circuit test", seica.com www.seica.com/technology/parametric-and-in-circuit-test/ (accessed Jun. 16 2020)

[3] *B.Thomas*, "Modern Reglerteknik", 5th ed., Stockholm: Liber, 2016

[4] *Sang-Hoon Kim*, "Electric Motor Control, DC, AC and BLDC Motors", Seoul: Elservier, 2017

[5] *S. Derammelaere, B. Vervisch, J. De Viaene, and K. Stockman*, "Sensorless load angle control for two-phase hybrid stepper motors," MECHATRONICS, vol. 43, pp. 6–17, 2017.

[6] *OpenLab*, "Stepper Motor Hookup Guide", accessed[May 31 2020] Available:https://openlabpro.com/guide/openlab-stepper-motor-hookup-guide/

[7] *Matthew Scarpino*, "Motors for Makers: A Guide to Steppers, Servos, and Other Electrical Machines", Indianapolis, Pearson Education, 2016

[8] *Zhongxue Gan, Qing Tang*, "Visual Sensing and its Applications: Integration of Laser Sensors to Industrial Robots", Zhejiang: Springer, 2011

[9] *Steve Maclean*, "Intelligent Load Cells and the Fieldbus" , InstMC, Volume 36, February 2003,[online]. Available: https://journals.sagepub.com/doi/10.1177/002029400303600102 [accessed Apr.24, 2020]

[10] *Davide Giovanelli. Elisabetta Farella*, Force Sensing Resistor and Evaluation of Technology for Wearable Body Pressure Sensing, Hindawi Publishing Corporation, [online]. Availble: $http://downloads.hindawi.com/journals/js/2016/9391850.pdf$. 2016, [accessed May 6 2020]

[11] *Betty Lincoln*, Digital electronics, 1st ed.,Noida: Pearson India, 2014

[12] *Arduino* , "What is Arduino?", Arduino, [online]. Availble: arduino.cc/en/Guide/introduction. [accessed Apr. 2, 2020].

[13] *Frederic Leens*, "An introduction to I^2C and SPI procotols", Mons: IEEE, 2009

[14] *Deepa Kaith, Janakkumar B. Pate, Neeraj Gupta* An Introduction to Functional Verification of I2C Protocol using UVM, Amity University Haryana, July 13, 2015, Accessed: May. 26, 2020.[online]. Available: Elseiver

[15] *Elfa*, "17H2A4417-Stegmotor", $https://www.elfa.se/sv/stegmotor-400nmm-nema-17-motionking-17h2a4417/p/30097243$ [accessed May. 5, 2020]

[16] *Can Stack Linear Actuators*, "42DBL-L datasheet", https://data.deltaline.com/media/42dbl_l_specifications_1188.pdf, V031519, 2019 [accessed May. 5 2020]

[17] *Velleman*, VMA409, 'L298N Dual Bridge DC Stepper Controller Board', V.01, 2017 [accessed May 18 2020]

[18] *Grove-Servo*, "ES08A datasheet"
Available: $https://www.elfa.se/Web/Downloads/_t/ds/316010005_engtds.pdf?pid=product.code$, 2015, [accessed May. 5 2020]

[19] *SG90 Servo datasheet*, Luxorpart,
Available: https://www.kjell.com/globalassets
/mediaassets/701917_90720_datasheet_en.pdf?ref=04B59C6757, [accessed 'May 30 2020]

[20] *FSR Series*, Ohmite, Elfa.se, rev2/18-2, [accessed May 3]

[21] *User manual for DM556T*, Stepper Online 2017, accessed Jun. 16 2020, [Online],
Available: www.stepper-store.com/wp-content/uploads/2019/02/DM556T.pdf

9 Appendix

A Arduino Mega code

```
#include <EdgeDebounceLite.h>
#include <Stepper.h>
#include "Wire.h"
#include "HX711.h"
#include "Adafruit_VL53L0X.h"
extern "C"{
  #include "utility/twi.h"
}

#define Ard 0x01
#define dSens1_adr 0x30
#define dSens2_adr 0x31

/*-------------------------------------------------------------------------------------
|                                    Variabler                                        |
-------------------------------------------------------------------------------------*/
//Arrays för sensorna
int spots[23]= {38,45,60,79,100,113,129,144,159,176,188,205,222,237,254,268,281,297,312,325,341,353,357}; //Distanser i rack 1
int spots2[23]={45,57,69,87,101,118,134,148,166,180,198,215,228,242,261,275,290,307,320,335,350,363,368}; // Distanser i rack 2

//Variabler för sensorerna
byte emptySlot;      //Antal tomma platser
const float calFac = 221.5; //Kalibreringsfaktor till vågen
byte xdown[2] ={13,12};
int d[20];            //Till distansmätning
int mean=0;           //Medeldistans
int sum=0;            //Summan av distansvektornskomposanter
int spot=0;          //Vilken plats
int numb=0;

//Stepositioner
int horisPos[23] = {1505, 1430, 1360,1290,1210,1110,1040, 970, 890,812, 730, 650, 580, 500, 420,340, 280, 200,120,7,7,7,7};

int horisPos2[23] = {2170,2250,2330,2410,2480,2560,2650,2710,2790,2850,2930,3010,3090,3160,3240,3310, 3390, 3460, 3540,3620, 3700,3770};
```

(a) 1

```
int startPos = 0;
int pickUpPos = 4350;
int prePickUp = 4000;
int goToPos;
int targetPos;
int stepCount=0;

byte ledPin[] = {38,39,40};
byte switchPin[2] = {30,31} ;
uint8_t Ard2 = 1;

EdgeDebounceLite debounce;
Stepper vertStep= Stepper(200, 26, 27, 28, 29);
Adafruit_VL53L0X dSens1 = Adafruit_VL53L0X();
Adafruit_VL53L0X dSens2 = Adafruit_VL53L0X();
VL53L0X_RangingMeasurementData_t meas1;
VL53L0X_RangingMeasurementData_t meas2;

//Axelns olika positioner
enum vertStates {START, WAIT, FRAM,PLOCK, BAK, FULLRACK};
vertStates nuvState = START;

//Knappens olika states
enum SwitchStates {Open, Risin, Closed, Fallin};
SwitchStates switchState[2] = {Open,Open};


enum SwitchModes {PULLUP, PULLDOWN};
SwitchModes switchMode[2]={PULLUP,PULLUP};
```

(b) 2

```
/*-------------------------------------------------------------------------------------
|                                    Knappar                                          |
-------------------------------------------------------------------------------------*/
//Maskinen som ändrar switchens läge
void switchMachine(byte i) {
  byte pinIs = debounce.pin(switchPin[i]);
  if (switchMode[i] == PULLUP) pinIs =!pinIs;
    switch (switchState[i]){
      case Open:    { pinIs=!pinIs;  if(pinIs==HIGH)  switchState[i]=Risin ; break;}
      case Risin:   {                    switchState[i]=Closed; break;}
      case Closed:  { pinIs=!pinIs;  if(pinIs==LOW)   switchState[i]=Fallin; break;}
      case Fallin:  {                    switchState[i]=Open  ; break;}
  }
}

//Funktion som läser av knapptryckningarna, kommer i vårt fall vara signal som mottages vid "färdig" aktion
bool testOk() {
  switchMachine(0);
    if (switchState[0] == Risin) return true; // Om knappen har klickats och sedan släppts får vi en true
    else                         return false;
}
bool testFail() {
  switchMachine(1);
    if (switchState[1] == Risin) return true; // Om knappen har klickats och sedan släppts får vi en true
    else                         return false;
}
```

(a) 3

```
/*-------------------------------------------------------------------------------------
|                                    Motor                                            |
-------------------------------------------------------------------------------------*/
void moveHor(int j){
  if (j<stepCount){
    while (stepCount>j){
      vertStep.step(-1);
      stepCount--;
      if (stepCount>=4200){
        vertStep.setSpeed(50);
      }
      else {
        vertStep.setSpeed(200);
      }
    }
  }
  else {
    while (stepCount<j){
      vertStep.step(1);
      stepCount++;
       if (stepCount>=4000){
        vertStep.setSpeed(50);
      }
      else {
        vertStep.setSpeed(200);
      }
    }
  }
}
```

(b) 4

```
void setId(){ //Satter iD på varje sensor
  for (int i=0;i<2;i++){
   digitalWrite(xdown[i], LOW);
}
delay(10);
  for(int i=0;i<2;i++){
    digitalWrite(xdown[i],HIGH);
  }
  delay(10);
  digitalWrite(xdown[0],HIGH);
  digitalWrite(xdown[1],LOW);
  if(!dSens1.begin(dSens1_adr)) {
    Serial.println(F("Failed to boot first VL53L0X"));
    while(1);
  }
  delay(10);

  digitalWrite(xdown[1],HIGH);
  delay(10);

  if(!dSens2.begin(dSens2_adr)) {
    Serial.println(F("Failed to boot second VL53L0X"));
    while(1);
  }
}
```

(a) 5

```
int nearPosition (int s, int A[]){
  int nearIndex=0;                  //Närmsta tomma slot från start
  int nearestDiff=abs(A[0]-s);      //Beräknar differensen mellan distansen och slot 0 i mm
   for(byte i = 0; i<23; i++){      //Går igenom hela array och beräknar differensen mellan distans s och slot i
    if(abs(A[i]-s)<nearestDiff){    //Om differensen är större än tidigare differens
     nearIndex = i;                 //Närmsta index blir då "i"
     nearestDiff = abs(A[i]-s);     //Närsmsta diff blir då diffen från position i till s
     }
   }
   if (nearIndex==0){
     return -1;
   }
  return nearIndex;
}

int distanceRead1(){
  sum =0;
  mean =0;
  spot=0;
  for (int i =0; i<10;i++){
    dSens1.rangingTest(&meas1,false);
    d[i] = meas1.RangeMilliMeter;
    sum = sum + d[i];
  }
  mean=sum/10;
  spot=nearPosition(mean, spots);
  return spot;
}

int distanceRead2(){
  sum =0;
  mean =0;
  spot=0;
  for (int i =0; i<10;i++){
    dSens2.rangingTest(&meas2,false);
    d[i]=meas2.RangeMilliMeter;
    sum = sum + d[i];
  }
  mean=sum/10;
  spot=nearPosition(mean, spots2);
  return spot;
}
```

(b) 6

```
byte infromA2;
void vertMove() {
  switch(nuvState){
    case START:{
     moveHor(0);
     byte noll=0;
      Serial.println("Card ok or not?");
     digitalWrite(ledPin[0],LOW);digitalWrite(ledPin[1],HIGH);digitalWrite(ledPin[2],LOW);

  if (testOk()){
    delay(100);
    digitalWrite(ledPin[0],HIGH);digitalWrite(ledPin[1],LOW);digitalWrite(ledPin[2],LOW);
    goToPos = distanceRead1();
    if (goToPos==-1){
      digitalWrite(ledPin[0],LOW);digitalWrite(ledPin[1],LOW);digitalWrite(ledPin[2],HIGH);
      nuvState=FULLRACK;
    }
    else{
    targetPos=horisPos[goToPos-1];
    Serial.print("Antal steg som måste tas tillbaka: ");Serial.println(targetPos);
    delay(100);
    nuvState=WAIT;
    }
  }
  else if (testFail()){
    digitalWrite(ledPin[0],LOW);digitalWrite(ledPin[1],LOW);digitalWrite(ledPin[2],HIGH);
    goToPos=distanceRead2();
    if(goToPos==-1){
       digitalWrite(ledPin[0],LOW);digitalWrite(ledPin[1],LOW);digitalWrite(ledPin[2],HIGH);
      nuvState=FULLRACK;
    }
    else{
     targetPos=horisPos2[goToPos-1];
    Serial.print("Target position: "); Serial.println(targetPos);
    delay(100);
    nuvState=WAIT;
}
  }
break;
    }
```

(a) 7

```
  case FULLRACK :{
    delay(20);
    Serial.println("Rack full");

    if(testOk()){
      delay(20);
      while (!testFail()){
          Serial.println("Rack changed? If yes, press red");

        }
        Serial.println("Process restart");
         nuvState=START;


      }
     break;
    }


  case WAIT:
  {
    byte noll =0;
//      Wire.beginTransmission(1);
//      Wire.write(noll);
//      Wire.endTransmission();

    if(testOk())
    { nuvState=FRAM;
    }
    break;
    }
```

(b) 8

```
case FRAM:
{
  byte ett =1;
  digitalWrite(ledPin[0],LOW);digitalWrite(ledPin[1],HIGH);digitalWrite(ledPin[2],LOW);
  moveHor(prePickUp);
  delay(10);
if (numb==0){
  Wire.beginTransmission(1);
  Wire.write(ett);
  Wire.endTransmission();
  numb++;
}
  byte infromA2;

  delay(10);

  Wire.requestFrom(1,1);

  while(Wire.available()){
    infromA2 = Wire.read();
  }

   if(infromA2==1){
    numb=0;
    nuvState=PLOCK;
  }

  break;
  }

  case PLOCK:{
    byte tva=2;
  digitalWrite(ledPin[0],LOW);digitalWrite(ledPin[1],HIGH);digitalWrite(ledPin[2],LOW);
    vertStep.setSpeed(100);
    moveHor(pickUpPos);


    if (numb==0){
    Wire.beginTransmission(1);
    Wire.write(tva);
    Wire.endTransmission();
```

(a) 9

```
  }
      delay(10);
  Wire.requestFrom(1,1);

  while(Wire.available()){
    infromA2 =Wire.read();
  }

  if(infromA2==2){
    numb=0;
    nuvState=BAK;
  }
  break;
 }

case BAK:
{
  byte tre=3;
 moveHor(targetPos);
 if (numb==0){
 Wire.beginTransmission(1);
 Wire.write(tre);
 Wire.endTransmission();
 numb++;
 }
 delay(10);
 Wire.requestFrom(1,1);
 while(Wire.available()){
  infromA2 =Wire.read();
 }

    if(infromA2==3){
    numb=0;
    nuvState=START;
    }
  break;
  }
 }
}
```

(b) 10

```
void setup() {
  // put your setup code here, to run once:
vertStep.setSpeed(200);
Wire.begin();
Serial.begin(115200);

for(byte i=0;i<2;i++){                          //Anger pinmodes för leds och switchar!
    pinMode(switchPin[i], INPUT_PULLUP);
    pinMode(xdown[i], OUTPUT);
    digitalWrite(xdown[i],LOW);
  }
  for (int i=0;i<3;i++){            //Sätter Output och läge för Leds
  pinMode(ledPin[i], OUTPUT);
    digitalWrite(ledPin[i],LOW);
  }
  setId();


}

void loop() {
vertMove();

}
```

Figure 17: 11

B Arduino Uno code

```
//Including neccesarry libraries
#include <Wire.h> //The I2C library
#include <Stepper.h>// The stepper library
#include <Adafruit_TiCoServo.h>// The servo library

#define PRESS A0 //Defines the input to the FSR
uint8_t adress = 1; //Adress of the slave

byte masterRec =0; //Value recevied from the master
byte masterTrans =0;//Value Transceived to the master

byte stateIteration = 0;//

enum states {WAIT, ANGLE, PICKUP, DROP}; //Enums for states in the statemachine
states state = WAIT;

Adafruit_TiCoServo servo1; //Creating a servo
int servoPos1; //a integer to know the position of servo 1
Adafruit_TiCoServo servo2; //Creating a  second servo
int servoPos2;//a integer to know the position of servo 2

int value = 0; //Sum of the values of the read values of the pressure sensor
int mean = 0; //mean value from the pressure sensor
int value2=0; //Value read from the pressure sensor
int pressArray[5]; //Array that contains mean values from the pressure sensor

const int STEPS = 48; //Steps per revolution for the Linear actuator
Stepper stepper(STEPS, 5, 6, 7, 8);
int stepCount =0; //Initiate a variable to know how far the stepper motor
//is from the starting point. The stepper has three different position.
```

(a) 1

```
//A method to be called when we want to recevie something from the master
void receiveEvent (int howMany){
  while(0<Wire.available()){
  masterRec = Wire.read(); //Reads data from the master
  Serial.print("Slave receives a ");
  Serial.print(masterRec); //Writes what it received
  Serial.println(" from the master");
  }
//  flag = HIGH;
  //return masterRec;
}
//A method to be called when we want to transceive something to the master
void requestEvent(){
  delay(1);
  Wire.write(masterTrans);//writes data to the master
  delay(100);
  }

//A method that will turn the servo to a special angle
void servoTurnTo (Adafruit_TiCoServo servo, int degree){
 int y = servo.read();//Servo read it position
 if (y!=degree){
 int x= 10*abs(servo.read()-degree);
 servo.write(degree);
 delay(x);
}
else{
  Serial.println("Servo already in position");
}
}
```

(b) 2

```
//Different array methods that is used in pressureMethod()
//A method that move every index one position and puts a new value in the first position
void putInArray(int arr[], int lastValue ){
  for( int i=1; i<sizeof(arr) ; i++){
    arr[i] =arr[i-1];
  }
  arr[0]= lastValue;
 }
 //A method that return the maximum value of the array
 int getMaxValue(int arr[]){
  int MAX = arr[0];
  for (int i =1; i<sizeof(arr); i++){
    if(MAX<arr[i]){
      MAX=arr[i];
    }
  }
return MAX;
}
//A method that return the minimum value of the array
int getMinValue(int arr[]){
  int MIN = arr[0];
  for (int i =1; i<sizeof(arr); i++){
    if(MIN>arr[i]){
      MIN=arr[i];
    }
  }
return MIN;
}
//A method that makes every position in the array 0
void arrayZero(int arr[]){
  for(int i =0; i<sizeof(arr); i++){
    arr[i]= 0;
  }
 }
```

(a) 3

```
// The original pressure measurement method
bool pressureMethod(){
 value=0;
value2=0;
mean=0;
  //Takes take measurement from the pressure sensor
  for(int i=0; i<10; i++){
  value2 = analogRead(PRESS);
  Serial.print(value2);
  Serial.print(", ");
  value += value2;
  delayMicroseconds(1);
  }
Serial.println("");
mean = (value/10); //The mean of the 10 measurement is taken
Serial.print("mean: ");
Serial.println(mean);
putInArray(pressArray, mean); //The mean is putted in to the array
int y = getMaxValue(pressArray); //Check what the maximum value of pressArray
int a= getMinValue(pressArray); // Check the minimum value of pressArray
int z = y-a; //checks the difference beetween the maximum and the minimum value
Serial.print("Difference in array: ");
Serial.println(z);
//A if statement that returns false if the minimum array is
if (a<400 || z>10) {
  return false;
}
else {
  return true;
}}
```

(b) 4

```
//A boolean method that checks if the PCB i gripped
bool gripCheck(){
  for(int i=0; i<10; i++){
  value2 = analogRead(PRESS);
  Serial.print(value2);
  Serial.print(", ");
  value += value2;
  delayMicroseconds(1);
   }
  mean= value/10;
  if (mean>100){
    return true;
  }
  else{
    return false;
  }
}
//A method that will turn a servo to 180 degrees and then check if it is gripped.
void pressureMethod2(Adafruit_TiCoServo servo, int servoPos){
  value=0;
  value2=0;
  mean=0;
  servoPos = servo.read()+1;
  for(servoPos; servoPos<180; servoPos++){
    servo.write(servoPos);
    delay(10);
  }
  bool x= gripCheck();
  if(x==true){
    Serial.println("PCB gripped");
  }
  else{
    Serial.println("Error, PCB not gripped");
  }
}
```

(a) 5

```
//A method that will make the servo take one degree step at the time until
//the pressure is reached or it has done a full 180 rotation
//This was the original method for gripping the PCB
void underPressure (Adafruit_TiCoServo servo, int servoPos){
  servoPos= servo.read();
  Serial.println("Position in degrees");
  while(pressureMethod()==false && servoPos<180){
     servoPos++;
     servo.write(servoPos);
     delay(5);
     Serial.println(servoPos);
  }
  }
//A method that makes the servo go to a position smooth
 void servoGoBack(Adafruit_TiCoServo servo, int degree){
  int x =servo.read();
  for( x; x>=degree; x--){
    servo.write(x);
    delay(10);
  }
  Serial.println("Servo in position");
 }
```

(b) 6

```
//Method that makes the linear actuator go to a certain position
void goToPosition (Stepper steg, int pos){

  Serial.print("Steps to be taken: ");
  Serial.println(pos);
  int y = 0;  int x = 0;
  //x is on which value of stepCount the position is in
  if (pos==1){
    x = -500;
  }
  else if (pos==2){
    x = -2100;
  }
  else if(pos==3){
    x= -1000;
  }
  else if( pos==4){
    x= -1500;
  }
  else{
    x=0;
  }
 //The amount of steps to be taken. A positive value will make the gripper move uppwards.
 //A negative number will make the gripper move downwards
  y = x-stepCount;
  if (y!=0){
  steg.step(y);
  stepCount+=y;
  delay(1000);
  }
```

(a) 7

```
void setup() {
  // put your setup code here, to run once:
Wire.begin(adress);
arrayZero(pressArray);
delay(10);
servo1.attach(10);//Servo one signal cable connected to digital 5
delay(10);
servo2.attach(9); //Servo two signal cable connected to digital 3
delay(10);
stepper.setSpeed(250); //speed of stepper motor
//stepper.step(10);
delay(1000);
//stepper.step(-10);
delay(1000);
delay(10);
servo1.write(35); //Servo to 0 degree
delay(500);
servo2.write(8);
delay(500);
//Wire.onRequest(requestEvent);
Serial.begin(115200);
//Will make the linear acuator go uppwards until a key is pressed on the computer
//Make it go to the top
while(!Serial.available()){
  stepper.step(1);
  delay(10);
}
//Puts the linear actuator down 450 in the initiate position
stepper.step(-450);
delay(1000);
goToPosition(stepper, 2);//Goes down to pick up position
Serial.println("Program begins");
delay(500);
}
```

(b) 8

```
void loop() {
  // put your main code here, to run repeatedly:
switch(state) {
 case WAIT:{
  //The first iteration
  if (stateIteration==0){
    goToPosition(stepper, 2);//Goes to pick up Position
    stateIteration++;
  }
  Wire.onReceive(receiveEvent); //Receives data from the master
  delay(1);
  if(masterRec==1){//If a 1 is received from the master
  delay(100);
  servoTurnTo(servo1, 135);//Servo turns to 90 degrees
  state= ANGLE;
  stateIteration=0;
  }
  break;
  }
  case ANGLE:{
 if(stateIteration==0){
  masterTrans=1;
  delay(1000);
  Wire.onRequest(requestEvent); //Sends value to master
  delay(1000);
  Wire.onReceive(receiveEvent);
  stateIteration++;
 }
 else{
  Wire.onReceive(receiveEvent);
 }
  if(masterRec==2){
    pressureMethod2(servo2, servoPos2);//The gripper starts grippin
    stateIteration=0;
    state= PICKUP;
  }
  break;
  }
  case PICKUP:{
  if(stateIteration==0){
  delay(10);
  masterTrans=2;
  goToPosition(stepper, 1);//Linear actuator goes up
  delay(10);
  Wire.onRequest(requestEvent);
  delay(1);
  Wire.onReceive(receiveEvent);
  stateIteration++;
  }
  else{
   Wire.onReceive(receiveEvent);;
  }
  if(masterRec==3){
    servoGoBack(servo1, 35);//Servo 1 goes back to 0 degrees
    stateIteration=0;
    state=DROP;
  }
  break;
  }

  case DROP:{
  goToPosition(stepper, 2);//Stepper goes down
  delay(1);
  servoGoBack(servo2, 8);//Gripper drops
  delay(1);
  goToPosition(stepper, 4);//Stepper goes upp a small bit
  //masterTrans=3;
  Serial.println("dropped");
```

(a) 9 (b) 10

```
  delay(10);
  masterTrans=3;
  Wire.onRequest(requestEvent);
  delay(1);
  state=WAIT;
  arrayZero(pressArray);
  break;
  }
}
}
```

Figure 18: 11

C Figures of system

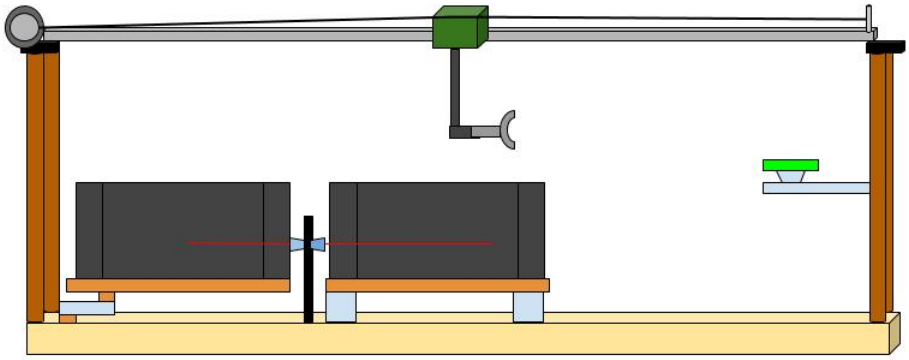


Figure 19: Sketch of the system

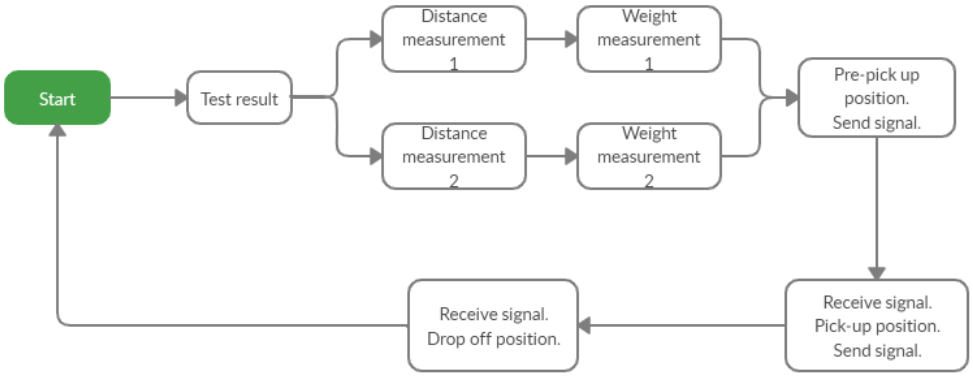


Figure 20: Mega process-scheme

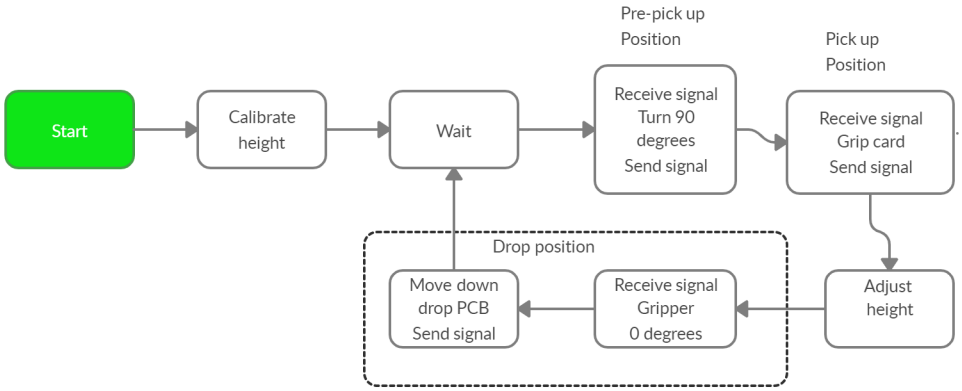


Figure 21: Uno process-scheme