The University of the Witwatersrand

ELEN 1002 – Concepts of Design

BSc Applied Computing

Project 2:

Elevator Prototype Model

2013

Group C: Jason Max Chalom 711985, Jonah Hooper 728237, Joshua Pike 721564

Date of Submission: Tuesday 20th May 2013

Mentors in Order: Dr. Stephen Levitt, Professor Rex van Olst, Professor Scott Hazelhurst.

Abstract:

The report describes a design and model for a vertically travelling freight elevator which is able to carry cargo three storeys within a building. The model was constrained to make use of scrap parts and an Arduino UNO with an ATMEGA324 microprocessor. The elevator must also have input buttons and some form of output to display critical information. The weight which the elevator is carrying is assumed to be negligible. The user is also assumed to press the button on the first floor for the direction in which the user wishes the elevator to go in. An initial prototype was built out of Lego but was found to be inadequate. The final model made use of parts found in an old inkjet printer and is entirely functional. The structure has the dimensions of 42cm x 20cm x 20xm. It is however not sturdy enough for real world application without some modification to its structure such as cross beams and the addition of an extra carriage guiding axel on its other side to stabilise the carriage. The motor uses 4.59 watts of energy to move the carriage vertically. The design is environmentally friendly and almost entirely recyclable. A new design with the additional modifications would be highly efficient and transport goods in a real world application very effectively.

Contents

1. Introduction

The task given was to design and build a prototype elevator to be used in a three storey building. The prototype must be built out of scrap components such as electronic devices found inside an old printer or raw material off cuts such as sheets of wood, metal or other such materials. The design must also make use of the ATMEGA324 microprocessor found in the Arduino Uno platform. The prototype is expected to move vertically while the weight of its load is considered negligible.

The user interface must incorporate call buttons on the exterior of the shaft for the user to be able to call the elevator the respective floor. There should also be buttons located allocated for the inside of the carriage to control the destination of the elevator. There must also be an indication of the floor the elevator is going to and the floor it is currently on.

The algorithm used in the operation of the elevator should be efficient and should be able to handle multiple inputs in an optimised way.

The elevator prototype is a design for a freight elevator to be placed either in the back of a building used for industry.

It is the aim for this prototype to be functional and reliable as well as of sturdy construction. It is desired to prove that the prototype will be viable to be used in a three storey building at full scale of the building.

2. Background

Elevators have been used for many centauries as an efficient form of vertical transportation within a shaft while saving space. The shaft can be built completely vertically saving on space which would otherwise be wasted building at an oblique angle for a more conveyer belt operated system [1].

Primitive elevators have been found to have been constructed from the third century BC but the earliest elevators to have been constructed to resemble the elevators used today were built in the 19th century during the Victorian and Gilded ages respectively and were operated by steam engines. In the later parts of the 19th century electric driven elevators began to be used because of their high torque ability which steam engines lacked. This high torque enabled these elevators to reach much greater heights faster and safer [1].

The basic operation of the elevator works on a very simple and very old mechanical device known as the hoist, pulley or winch system. This mechanical system enables a heavy load to be pulled too much greater heights with not as much effort and work than would be conventionally used without using these mechanical devices [2].

2.1 Current Solutions

There are many different electric elevator models and designs available since it is a technology which is widely used all around the world for construction and demonstration purposes. There are innumerable amount of hobbyist prototypes available on the internet. The approaches to creating these designs vary from the construction to the mechanics and programming involved in the elevators operation [3].

These designs make use of many different materials such as Mechano, Lego, Cardboard and even wood. Most designs have a counterweight system in place to steady the elevator, enable smooth operation and reduce the load on the gearing systems which provide the kinetic energy to lift the carriage within the shaft. They are mainly aimed at showing the microcontroller and programming of the elevator rather than the design. The main types of microcontrollers used are the Arduino and the LEGO Mindstorm platforms [3].

An example of a professional scale model of a real world elevator would be the main high speed elevator used in Taipei 101. This model unlike the hobbyist ones is perfectly scale and has small versions of the same mechanisms actually used in the real life elevator. The scale model is also well constructed from specifically engineered parts. This model is used to show the advanced technologies used in the elevator such as its ceramic braking mechanism [4].

2.2 Equipment

Many construction tools, such as screwdrivers, saws, setsquares, and clamps were used in the construction of the model.

A laptop computer with the Arduino software development kit to program and monitor the Arduino UNO while it is in operation controlling the elevator.

The equipment in the laboratory used to assess the design of the prototype were digital cameras to document the progress of the design and a multi-meter which was used to check the circuits current and voltage ratings as well as continuity testing to check for breaks in the connections.

An external power source is also used to power the motor and the laptop is used via the universal serial bus (USB) cable to power the Arduino UNO.

2.3 Constraints and Assumptions

The constraints provided are that the prototype carriage must be able to travel three storeys, have a user interface so that a user may call the carriage to the appropriate floor and also send it to the appropriate floor. The elevator must also operate smoothly and the prototype must be made of scrap materials.

It is assumed that the elevator will only carry a negligible load weight and that the system will at most have two users operating it at a time.

It is assumed that when a user calls an exterior middle button, they intend to go to the floor in the corresponding direction.

3. Design

The design proved challenging where the prototype had to be designed around the scrap materials that were available for use. This proved challenging where the scarp materials in some cases had to be altered to work as part of a vertical elevator.

Many components used in the main functionality of the model were taken from a Hewlett Packard Deskjet Printer such as the main direct current (DC) electric motor, the drive belt, the inkjet head carriage and the main drive axel. Section 3.2 discusses the different parts and materials used.

An initial prototype made out of LEGO was proposed and a scale model was built but it was found to be inadequate for a final elevator prototype which is discussed in section 3.4.

The final design was made to be stable yet simple in construction so that the focus of the construction and testing would be on producing a working model.

3.1 Requirements

The structure is required to be stable and ridged while also not obstructing the mechanics and operation of the actual elevator. It must also be able to hold the electric DC motor and metal drive axel vertically so that the elevator may be able to ride on the metal drive axel which will act as a support and stabiliser. The design must also follow the constraints provided by the project brief and discussed in section 1 and 2.3.

3.2 Supply of Parts Used in the Construction of Both Prototypes

The project was designed so that scrap materials which there was access to could be exploited such as LEGO, printer parts, wood, screws and heat glue.

Jonah had access to old LEGO parts which was used in both prototypes and Jason had access to left over sheets of wood, screws, heat glue and tools used in past high school projects. A more comprehensive list of parts used can be found in Appendix B.

3.3 Early Development Ideas

During the brainstorming stage of the project a couple of different ideas were thought of to elevate the carriage in the shaft.

One was the use of a jackscrew which could then also stabilise the carriage and it would also be extremely precise at moving the carriage between floors. These revolutions could then be calculated to the number of windings the screw had. It would be of similar design to how Makerbot three-dimensional printers operate [5]. The idea was not implemented because its construction was considered too complex due to the perceived difficulty of creating the parts for a screw system and because it did not lend itself to the already available scrap material that was on hand.

Another idea involved using pulleys to wind up a spool of nylon string when the elevator vertically ascended and then unwinding the spool to allow gravity to pull the carriage down when the elevator was needed to descend. This idea was not implemented due to many problems which may be caused by this system. The string may ‘bunch up’ within the spool thereby jamming the system and making the carriage stick. It would also not be a very responsive system since the carriage is carrying a negligible load and the gravity acting upon the carriage would therefore not be effective in pulling the carriage down and acting against the friction of the gearing. Another problem is that there is a simpler solution to moving the carriage vertically in the shaft. This is the solution used in the final design. Here no extra gears besides the motor itself, the drive belt from the printer and an old bobbin were used. This is discussed later in section 3.5.

3.4 Initial Prototype

The initial prototype (see Appendix D Figure 1A), was a concept design which was used to contextualise the ideas for used in the final prototype. The design was quite flawed as it was small in diameter, only having a length of 6.5cm, a width of 4.5cm and a height of 15cm. This did not meet our requirements and an actual carriage which would fit in the shaft would be difficult to build and realistically small and solid core since Lego bricks are of certain specifications where every two dots on each brick are 17mm in length. This means that any carriage that is built would be solid.

As seen in the photographs of the model (see Appendix D Figure 1B), there is also no place to install the electronics, the motor or the gearing system and this prototype does not integrate well with the recycled parts such as the main drive axel of the printer which is much taller than the model since it is 41 centimetres (cm) long. There was also not enough Lego available to produce a large enough model for it to be functional.

3.5 Final Prototype

The final prototype design (see Appendix D Figure 2A) makes use of the basic system which the Hewlett Packard Deskjet 5652 printer used to move the print heads over the page. This system was found to work by inspection as having a carriage which is designed to clip onto a thick metal bar or drive axel, and is then pulled along the length of this axel by a drive belt, gears and an electric motor.

The design follows a basic bird cage structure which enables it to be adequately robust while not being too complicated as to interfere with the mechanics of the actual elevator system [4].

It has the basic structural dimensions of a square base which is 20cm by 20cm and a vertical height of 42cm. The design must be taller than the drive axel so that the axel will fit neatly onto the design. It has specific bevelled holes through its core to allow the bar to be affixed to a stand or in the case of the prototype for it to be affixed to a corner pillar of the model.

The design has a shelf underneath the roof which houses the motor. This is an easy way of securing the motor to the model by having supports for the motor on three sides since the motor which was recovered only has treaded holes in the front. The motor was secured in place with heat glue.

An old bobbin was used as the bottom pulley since it does not provide much friction for the belt and it acts as a static pulley to allow the belt to rotate freely.

The time allocated to this project was divided up into the many design, construction and testing stages which are tabulated in Appendix A Section 1. The work was divided equally amongst the group members which is also in Appendix A under Section 2.

3.5.1 Material and Parts

The materials required to be used in this project had to be scrap materials and a list of used materials can be found in Appendix B.

The model was built primarily out of wood with the carriage made out of Lego. Wood was used since it is a relatively easy material to craft and the group members have had previous experience working with it in high school. Lego was used to build the carriage because it was available and is an easy material to use to quickly build and then dismantle to make changes to the design.

The design makes use of an H-Bridge Integrated Circuit(IC) which was decided to be used rather than a set of relays or a set of metal-oxide-semiconductor field-effect transistors (MOSFETs) was because the IC is much more responsive than the relays and also uses less power than both the relays and the MOSFETs. It also operates at a cooler temperature than the MOSFETs. This makes it possible for the Arduino to power all the Light Emitting Diodes (LEDs) and the IC on one circuit. See Appendix B Section 2.2 for more documentation on the H-Bridge.

3.5.2 Mechanics

The design makes use of a very basic mechanical system. It consists of a drive belt, a motor and a static pulley which gives no friction to the belt which allows it to move freely. The static pulley is an old metal bobbin. The motor is attached to the model above the three floors and under the roof of the structure on a shelf which has been attached to the roof. The bobbin is glued near the base of the model on the pillar which is vertically in line with the driver of the electric motor. The belt is then placed on the motor and pulled taught and attached to the bobbin.

This then allows the motor to move the belt in a circular direction. To make the belt move the plastic carriage it is glued to the main part of the carriage which is the printer cartridge carriage which was modified for the elevator model. See Appendix B Section 1 for more information about the parts used.

The electric motor is small enough to stop almost instantly when no power is given to it. This means that neither a complicated gear box nor brakes are needed to stop the elevator.

3.5.3 Control

The model is controlled by a series of switches both for the exterior of the elevator shaft on each floor and on the interior of the carriage of the elevator. These buttons connect to the microprocessor which then interpret the inputted button presses and move the elevator accordingly. See section 3.5.5 for the algorithm of how the elevator operates. The Arduino also gets inputs from a series of contact switches which are located at each floor so that when the elevator reaches a floor the Arduino can cut the power to the motor making it stop. There is also an emergency stop button which when pressed will make the Arduino stop the elevator at the location it is at.

3.5.4 Electronics

The design of the circuit is fairly basic (See Appendix B Section 2.1 for the circuit diagram). As little amount of parts as possible were used. The entire system truly contains two major components which are the Arduino UNO and the H-Bridge Controller. The Arduino controls the entire Elevator while the H-Bridge Controller controls the electric motor for the Arduino. In section 3.5.1there is an explanation for the use of an Integrated H-Bridge solution. There are 8 tactile buttons used for input and 8 LEDs used for output. There are resistors connected to every button to eliminate floating voltage issues which were encountered. No resistors were required for the LEDs since they are powered by the +5 Volt (V) rail from the Arduino which has an acceptable resistance for running LEDs without spiking them.

3.5.5 Algorithm and Code

The elevator algorithm was designed to move the elevator carriage up and down a three story elevator shaft based on the state of the overall elevator system. This movement was intended to be completed in an "optimised" manner, i.e. it should stop as few times as possible on its journey.

The elevators state consists of eight Boolean flags and one integer. One flag represents the direction of the elevator, three represent the input nodes within the elevator and the remaining four represent the exterior input nodes. The integer value represents the floor at which the elevator was last at rest or moved passed. This is a simple state machine which eliminates the need for a true queuing system. This unfortunately makes this algorithm quite greedy but as it runs in a infinite loop where no other computation occurs, this algorithm is optimal for its use as an elevator control algorithm.

There are three storeys in the elevator and there are four exterior buttons with the two in the middle representing an 'up' and 'down' call. The algorithm also assumes that when a user calls an exterior middle button, they intend to go to the floor in the corresponding direction.

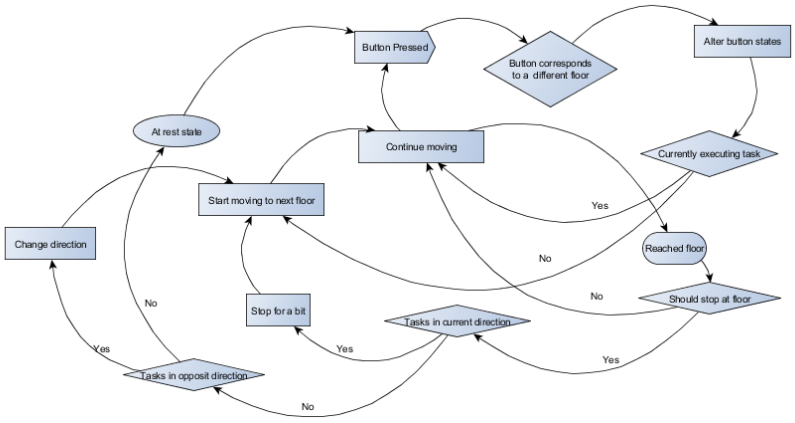
User input is primarily structured around calling the elevator from a floor and selecting the destination floor once inside the elevator. The elevator system can accommodate the input of multiple users on different stories, inside and outside the elevator. When the user presses a button the value of the corresponding input flag is changed accordingly. A flag will only change if the button selected is related to a different floor and has not been pressed before. Two of the four exterior button flags represent the elevators middle floor exterior input nodes. One represents a call to go the direction up and the other represents a call to go down.

The elevator will always start at a rest state. The program will then continuously scan for button inputs. When a button input is found the program will then verify that the button pressed corresponds to a different floor to the one at which it is currently resting. If the button corresponds to a different floor it will begin to move towards the corresponding floor. When the elevator is moving, additional button presses may be detected. These will alter the state of the input and thus influence which floor the elevator will move to. When it is detected that the elevator has reached any floor the algorithm will determine whether it should stop at that floor. The elevator will stop at a floor if the following conditions are met:

- A related interior button has been pressed
- An exterior button has been pressed that corresponds the same as its current direction
- The elevator can no longer physically continue in the current direction, i.e. it has reached the top or bottom floor.
- There are no further floors to visit in the current direction.

If none of those conditions are met, the elevator will continue moving; otherwise the elevator will stop at that floor and set its corresponding input states to false. If the elevator still has tasks to execute in the current direction, it will hesitate (to allow people to get in) and move on to the next floor. Otherwise, it will check that there are other tasks to do in the opposite direction. If there are tasks in the opposite direction it will change direction and move to appropriate floors. The algorithm will repeat the same steps as before, for the new direction. If it does not find another task in the opposing direction, the elevator will then stop moving and wait for new button inputs. See Appendix C Section 1 for source code used in the final model.

A simple flow chart diagram of the algorithm is shown on the next page to illustrate how the algorithm functions.



3.5.6 Environmental Concerns

The model is environmentally friendly because it is made of recycled materials and so it has a very low carbon foot-print. It may use some electricity to operate but the motor only uses 4.59 Watts of power (See Appendix B Section 2.1).

The wood, Arduino, Lego, electric motor, electronics and even the metal bar can be reused in other projects which lends this model to being itself highly recyclable.

There will be some slight waste due to the heat glue being not practical to reuse and the plastic carriage which was the cartridge holder in the printer originally having to be thrown out.

The use of breadboards means that all the electronics can be reused in other projects.

3.5.7 Expectations

It is expected that the model will be able to function so that the elevator may move and stop on each floor of the model. It must also be able to handle multiple inputs from more than one user and be able to display to the user what the elevator is currently doing such as moving and to what floor. The model must also be robust and be able to detect when the elevator reaches each floor.

4. Analysis of Design

The design does perform to the specifications and expectations set forward in sections 3.1 and 3.5.7. It is able to move up and down vertically and also it is able to stop at floors and handle multiple inputs by prioritising its current task and then performing the next task which would be travelling to the closest floor in the current direction the elevator is moving. The elevator also lights up certain LEDs to indicate when the motor is in operation. Other LEDs are also lit up to give the user vital information about the operation of the elevator. A red LED on a specific floor will come on when the elevator is moving towards that floor and a white LED will come on when the elevator arrives at that specific floor. The elevator is also able to successfully detect when it has reached a floor.

The design however is not truly robust. The structure is slightly skew and not completely level as seen in Appendix D Section 2 Figure 2B. The contact switches which are used to detect the different floors of the elevator are also flimsy and prone to breaking or bending out of shape.

Another problem with the model is that when the elevator is going down it does not have a smooth motion but rather the elevator moves down with a slight jitter. The elevator also moves too quickly.

4.1 Possible Improvements

Wood may not have been to strongest or most ridged material to use. Perhaps a type of metal or plastic would have been more ideal to use to build the structure out of. Also the use of electric power tools may have increased the accuracy at which the wood was cut thereby enabling a more accurate set of parts which fit together better.

The elevator needs another guiding axel on the other side of the carriage which is the same as the one which is currently on the model but this one would be loose. It would serve to help keep the carriage level to the ground rather than leaning thereby stopping the increase of friction between the carriage and the drive axel. The addition of oil on the axel would also help alleviate this problem.

The speed problem could be fixed by either adding a resistor to the motor to reduce its power output or the input voltage to the H-Bridge controller could be slightly reduced.

The stability of the structure could also be increased by the addition of crossbeams [4].

5. Conclusion

The design successfully functioned and was able to meet most the expectations and constraints imposed on it. Unfortunately the design was not robust or sturdy enough for it to be considered to be viable to be used in a three storey building at the full scale of the building. There are a few improvements to the design which need to be done in order to make it viable for real world use. The design however is environmentally friendly and user interactive which makes the design in some respects ready for real world use when these proposed improvements have been added to the design. The functionality of the elevator is efficient and smart. When a new design is implemented it will help transport goods within the building effectively and with high efficiency.

References

[1] Bellis, A, ‘History of the Elevator’, About.com Guide, http://inventors.about.com/od/estartinventions/a/Elevator.htm, Accessed: 2013-05-12.

[2] Mitsubishi Electric, ‘History of the Elevator’, 2012, http://www.mitsubishielectric.com/elevator/overview/elevators/history.html, Accessed: 2013-05-12.

[3] Youtube.com, ‘Elevator Prototype Search’, http://www.youtube.com/results?search_query=Elevator+prototype&oq=Elevator+prototype&gs_l=youtube.3...15183.21307.0.21596.18.18.0.0.0.0.632.3521.3j5j5j1j1j1.16.0...0.0...1ac.1.11.youtube.iO2P-3auQKk, Accessed: 2013-05-11.

[4] National Geographic, Richard Hammond's Engineering Connections Season 1 Episode 2: ‘Taipei Tower’, Presented by Richard Hammond, 2008.

[5] Makerbot Industries LLC, ‘The Replicator’, http://store.makerbot.com/replicator-404.html, Accessed: 2013-05-12.

[6] Miller Collin. T, ‘L293DNE H-Bridge’, Aspirations of a Software Developer, April 2010, http://developeraspirations.wordpress.com/2010/04/19/l293dne-h-bridge/., Accessed: 2013-05-03

Appendix A: Time Management and Teamwork Report

1. Part 1 Timeline of Development

20 April - 29 April

- Initial Planning phase. Brainstormed ideas, researched important concepts like Arduino programming and motor control systems and created designs

29 April - 1 May

- Tested Arduino programming techniques, detirmined how electronic components functioned, designed curcuits, wrote Arduino sketches and worked on user input.

1 May

- Stripped down printer to salvage parts. Perfected user input capturing techniques by testing buttons in conjunction with LED's.

1 May - 6 May

- Tested motor control curcuit with H-bridge and Arduino contollers. Further tested user input capturing. Tested Arduino output to LEDs

6 May - 10 May

- Tested a simple elevator algorithm with a user interface and small motor and push button to substitute for floor detection sensor. Final algorithm written in time period.

11 May

- Finalised design of structure and actual elevator system. Built elevator stucture out of wood. Combined salvaged printer parts with structure

12 May

- Tested and finalised electronic system and final algorithm logic. Ensured that buttons, motors, curcuitry where all in working order. Extensive debugging was performed. Made further additions to elevator structure.

13 May

- Added floor detection system to structure. Integrated electronics and movement system. Build and attatched carriage. Performed final tests with elevator movement. More debugging.

14 May

- More Testing performed. Fixed design some design issues and movement issues. Attatched display LED's. Added cardboard floors.

15 May

- Presentation performed.

2. Part 2 Teamwork Report

The group divided the appropriate work equally between each member. Jonah was in charge of programming the microcontroller, designing the algorithm and helping with the building of the model. Joshua was in charge of helping to design and build the electronic circuits and construction of the prototype. Jason was in charge of helping to design and build the electronics the design and construction of the prototype and editing the final report.

Appendix B List of Materials, Parts and Circuits Used

1. Materials Used

The primary constraint of this project was that scrap materials must be used for everything but the electronics.

Wood was recovered from old high school projects and used to build the structure of the model. This included hardboard for the shelf, seven ply wood for the base and roof and corner cut pine wood for the pillars.

Lego was used since it was an available resource which had been kept as an old toy. It has proven to be useful in rapid prototyping

The mechanics used in the model were recycled from an old broken Hewlett Packard Deskjet 5652. The main 9V DC motor, the drive belt, the main drive axel and print head carriage were taken from the printer. The DC motor was manufactured by the Mitsumi Corporation and is rated at 9V DC but has no model or part number. The print head carriage was manufactured by the Hewlett Packard Company.

The print head carriage was modified by cutting off the arms which held the print cartridges, and it was smoothed down so that all was left was the clip for the main drive axel and a smooth flat piece of plastic coming out at a right-angle to the clip.

The DC Motor has a diameter of 35mm and a height of 60mm. At normal operation the motor drew 0.54 Amperes (A) of current as was shown on the external power source.

Therefore the power output of the motor can be calculated:

$$P = VI$$

$$\therefore P = (8.5)(0.54)$$

$$\therefore P = 4.59\ Watts$$

An old metal bobbin was used in the model and was found discarded in an old toolkit. It has a diameter of 20mm and a height of 15mm.

Some single core insulated wire was used to connect the electronics and breadboards together and also is used as contact switches to detect each floor of the model.

Two breadboards are used. The one is for the input to the Arduino UNO and the other is set up to handle the output of the Arduino to the motor and LEDs.

2. Electronics

2.1 Electronic Diagram

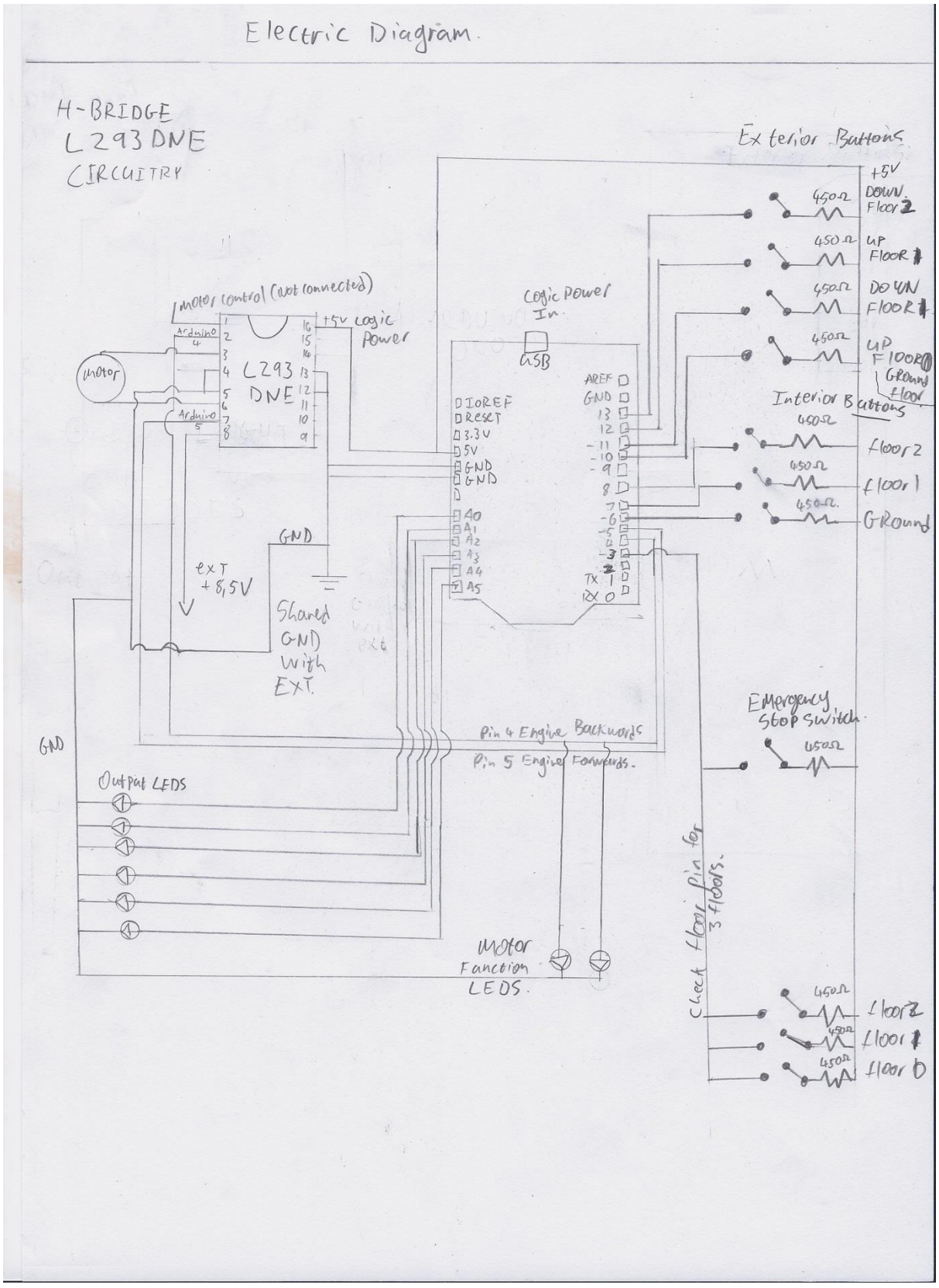


Figure 1

2.2 Electric Components

The main component used is one Arduino UNO microcontroller kit. This kit contains the ATMEGA324 microprocessor.

The main DC motor controller is a dual H-Bridge Integrated circuit (IC). It enables the microcontroller which only supplies a most a five volt rail with about 500 milliamps of current, to control a motor connected to a much higher power source. The IC used is the L293DNE Quadruple Half- H Driver from Texas Instruments, which is which is rated from 4.5 to 36 volts (V) with a maximum current of 1 Amp [6].

5 Standard Red and 3 White Light Emitting Diodes (LEDs) as well as 8 standard tactile push button switches have been used in the model.

Eleven 450ohm resistors have been used to eliminate floating voltages throughout the circuitry so that the buttons react more effectively.

A single 100 Nano-Farad capacitor is used across the terminals of the motor to stop reverse voltage leakage from occurring and shorting the internal circuitry of the H-Bridge. This is an extra layer of protection since the H-Bridge itself contains protection diodes [6].

Appendix C: Microprocessor Code

1. Arduino Source Code

```
/**SUMMARY**
    * The elevator program waits until a button is selected. When a button
is selected the elevator will move towards its associated floor

    * The elevator will visit all floors that have been called in its
current direction before changing direction

    * If more than one button is called in the same direction, the elevator
will visit either the highest or the lowest
      of the buttons that have been selected depending on its current
direction first before changing direction and executing the current task.
      For EXAMPLE, if the top button is called, and the second story going
down button is called, the elevator will skip the second story and proceed
to the top floor
      and then stop at the second story on its way down.

    * The elevevator will stop always stop at a story where a corresponding
interiorButton (Button inside the elevator) is selected

    * The elevator will stop at the top and bottom stories
*/
const int exteriorButtonPins[] = {10,11,12,13};
const int interiorButtonPins[] = {6,7,8};

//Set to true when the associated interior or exterior buttons are selected
const int NUM_INTERIOR = 3;
const int NUM_EXTERIOR = 4;
boolean interiorSelected[] = {false,false,false};
boolean exteriorSelected[] = {false,false,false,false};
```

```
const int NUM_LEDS = 6;
const int outputLEDPins[] = {A0, A1, A2, A3, A4, A5};

const int engineForwardPin = 5;
const int engineBackwardPin = 4;

const int checkFloorPin = 3;

int currentFloor = 0;

const int MAX_FLOOR_INDEX = 2;
const int MIN_FLOOR_INDEX = 0;
const int NUM_FLOORS = 3;

boolean currentDirection = true; //Stores the current direction the
elevator is travelling. true is up. false is down
boolean halt = true; //True when elevator is stationary

boolean floorDetectionLock = true;
unsigned long timeAtUnlocked;
const unsigned long MAX_BETWEEN_FLOOR_SENSED = 1000;
void setup () {
    Serial.begin (9600);
    setupLEDPins ();
    setupInteriorButtons ();
    setupExteriorButtons ();
    pinMode(engineForwardPin, OUTPUT);
    pinMode(engineBackwardPin, OUTPUT);
}
void loop () {
   /* if (DEBUG) {
        currentFloor = Serial.read ();
        DEBUG = false;
    }*/
    if (!halt) {
        if (hasReachedFloor()) {
                floorReachedEvent();
                if (shouldStopAtFloor (currentFloor, currentDirection)) {
                    Serial.print("Stopped at floor : " );
                    Serial.println (currentFloor);
                    stopElevator ();
                    delay (2000);
                    markFloorAsVisited(currentFloor);
                    Serial.println ("********************");
                }
                executeTask();
        }
    }
    if (buttonPressDetected())
            executeTask ();
    updateLEDs ();
}
/**
    Gets the elevator to start moving towards the next floor it should
visit
*/
void executeTask(){
    if (taskInDirection(currentDirection))
          startElevator ();
    else if (taskInDirection (!currentDirection)) {
```

```
          changeDirection();
          startElevator();
   }
}
/**
  Returns true if there is still a floor to visit in direction specified
*/
boolean taskInDirection(boolean pcurrentDirection) {
    for (int i = currentFloor; pcurrentDirection ? i < 3 : i >= 0; i =
pcurrentDirection ? i + 1 : i - 1) //Muhahahahahahahahaha
    {
        if (shouldStopAtFloor(i,pcurrentDirection))
          return true;
    }
    return false;
}
void setupExteriorButtons () {
    for (int i = 0; i < NUM_EXTERIOR; i++)
        pinMode (exteriorButtonPins[i], INPUT);
}
void setupInteriorButtons () {
    for (int i = 0 ; i < NUM_INTERIOR; i++)
        pinMode (interiorButtonPins[i], INPUT);
}
void setupLEDPins () {
    for (int i = 0 ; i < NUM_LEDS; i++)
        pinMode (outputLEDPins[i], OUTPUT);
}
//Returns true if a button press is detected
boolean buttonPressDetected () {

    return (readExteriorButtonStates ()
           || readInteriorButtonStates ());

}
boolean readExteriorButtonStates () {
    int count = 0;
    for (int i = 0 ; i < 4; i++) {
            if (count != getFloorForExteriorButton(i))
                  count++;
            if (digitalRead (exteriorButtonPins[i]) == HIGH && count !=
currentFloor) {
                exteriorSelected[i] = true;
                Serial.print ("Read exterior button ");
                Serial.println (i);
                return true;
            }

    }
    return false;
}
boolean hasReachedFloor () {
    if (digitalRead (checkFloorPin) && !floorDetectionLock && (millis() -
timeAtUnlocked > MAX_BETWEEN_FLOOR_SENSED)) {
        Serial.println("Detected reached floor");
        //delay(100);
        return true;
    }
    return false;
}
/**
```

```
    Digital reads all of the buttons and updates the interiorSelected flags
if a button is pressed
*/
boolean readInteriorButtonStates  () {
    for (int i = 0 ; i < 3; i++)  {
        if (i != currentFloor) {
            if (digitalRead(interiorButtonPins[i]) && !interiorSelected[i])
{
                interiorSelected[i] = true;
                Serial.print ("Read interior button ");
                Serial.println (i);
                return true;
            }
        }
    }
    return false;
}
/**
  Returns true if an interior button selection for the floor at index
specified was made or an
  exterior button that is in the same direction as the current floors
direction has been selected.
*/
boolean shouldStopAtFloor (int floorIndex, boolean pcurrentDirection) {
     switch (floorIndex) {
         case 0:
             return shouldStopAtFloorZero(pcurrentDirection);
         case 1:
             return shouldStopAtFloorOne(pcurrentDirection);
         default:
              return shouldStopAtFloorTwo (pcurrentDirection);
     }
}
boolean shouldStopAtFloorZero (boolean pcurrentDirection) {
    return (!pcurrentDirection && exteriorSelected[0])
||interiorSelected[0];
}
boolean shouldStopAtFloorOne (boolean pcurrentDirection) {
    return (pcurrentDirection && exteriorSelected[2])
              || (!pcurrentDirection && exteriorSelected[1])
              || (
                     (exteriorSelected[1] || exteriorSelected[2])
                           && (pcurrentDirection ?
!shouldStopAtFloorTwo(pcurrentDirection) :
!shouldStopAtFloorZero(pcurrentDirection))
                  )
              || interiorSelected[1];
}
boolean shouldStopAtFloorTwo (boolean pcurrentDirection) {
    return (pcurrentDirection && exteriorSelected[3]) ||
interiorSelected[2];
}
/**
    Increments or decrements floorReached variable depending on direction
of the elevator
    Ensures that the floor is marked as visited by deactivating its floor
visited commands
    Returns false if the elevator is at the top or bottomFloor
*/
boolean floorReachedEvent() {
    boolean succesfull = true;
```

```
    floorDetectionLock = true;
    if (currentDirection) {
          if (currentFloor < MAX_FLOOR_INDEX)
              currentFloor++;
          else {
              Serial.println("Reached top floor!");
              succesfull = false;
          }
      } else {
          if (currentFloor > MIN_FLOOR_INDEX)
              currentFloor--;
          else {
              Serial.println("Reached bottom floor");
              succesfull = false;
          }
    }
    return succesfull;
}
void markFloorAsVisited (int floorIndex) {
    interiorSelected[floorIndex] = false;
    switch (floorIndex) {
        case 0:
          exteriorSelected[0] = false;
        break;
        case 1:
           if (currentDirection)
               exteriorSelected[2] = false;
           else
               exteriorSelected[1] = false;
            if (!shouldStopAtFloorTwo(currentDirection) &&
!shouldStopAtFloorZero(currentDirection))
            {
                 exteriorSelected[2] = false;
                 exteriorSelected[1] = false;
            }
        break;
        case 2:
           exteriorSelected[3] = false;
        break;
    }
}
void changeDirection () {
      Serial.print("Changed direction to: " );
      Serial.println (!currentDirection);
      currentDirection = !currentDirection;
      digitalWrite (engineForwardPin, !currentDirection);
      digitalWrite (engineBackwardPin, currentDirection);
}
void stopElevator () {
      Serial.print("Did stop at floor: " ) ;
      Serial.println (currentFloor);
      digitalWrite (engineForwardPin,LOW);
      digitalWrite (engineBackwardPin, LOW);
      halt = true;
}
void startElevator () {
      Serial.print("Started from floor: ");
      Serial.println (currentFloor);
      digitalWrite (engineForwardPin, currentDirection);
      digitalWrite (engineBackwardPin, !currentDirection);
      halt = false;
```

```
      floorDetectionLock = false;
      timeAtUnlocked = millis();
}
void updateLEDs () {
     for (int i = 0; i < NUM_FLOORS; i++) {
         if (shouldStopAtFloor(i, currentDirection))
               digitalWrite (outputLEDPins[i + NUM_FLOORS], HIGH);
         else
               digitalWrite (outputLEDPins [i + NUM_FLOORS],LOW);
         if (i == currentFloor)
               digitalWrite (outputLEDPins[i], HIGH);
          else
               digitalWrite (outputLEDPins[i], LOW);

     }
}
void printExteriorButtonStates () {
      for (int i = 0 ; i < NUM_EXTERIOR; i++) {
            Serial.print ("Exterior Button : ");
            Serial.print (i);
            Serial.println (exteriorSelected[i] ? " TRUE " : " FALSE ");
      }
}
void printInteriorButtonStates () {
    for (int i = 0 ; i < NUM_INTERIOR; i++) {
            Serial.print ("Interior Button : ");
            Serial.print (i);
            Serial.println (interiorSelected[i] ? " TRUE " : " FALSE ");
      }
}
int getFloorForExteriorButton (int buttonIndex) {
      switch (buttonIndex) {
           case 0: return 0;
           case 3: return 2;
           default: return 1;
      }
}
```

Appendix D: Technical Diagrams and Photographs

1. Initial Prototype

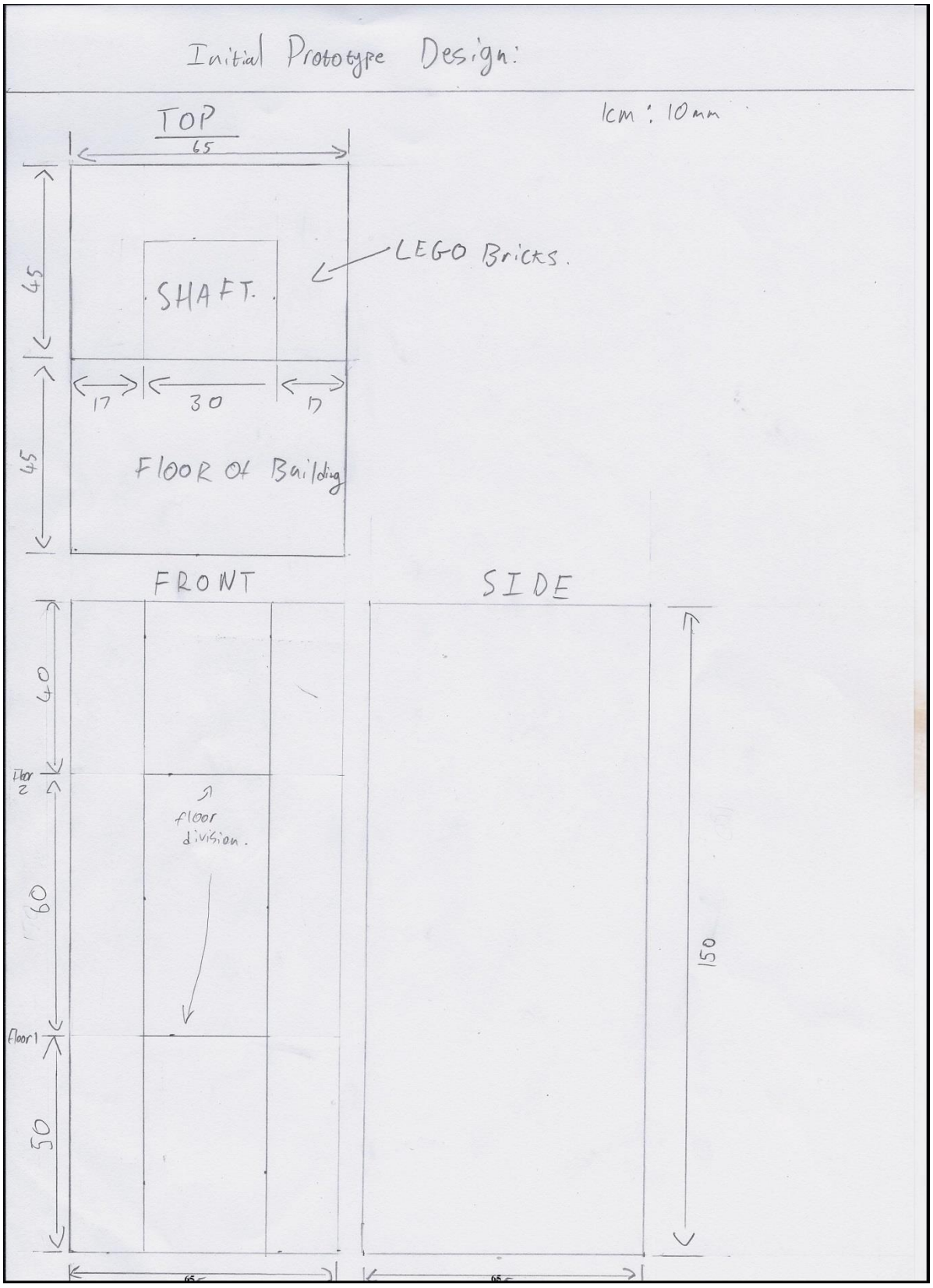


Figure 1A - Design

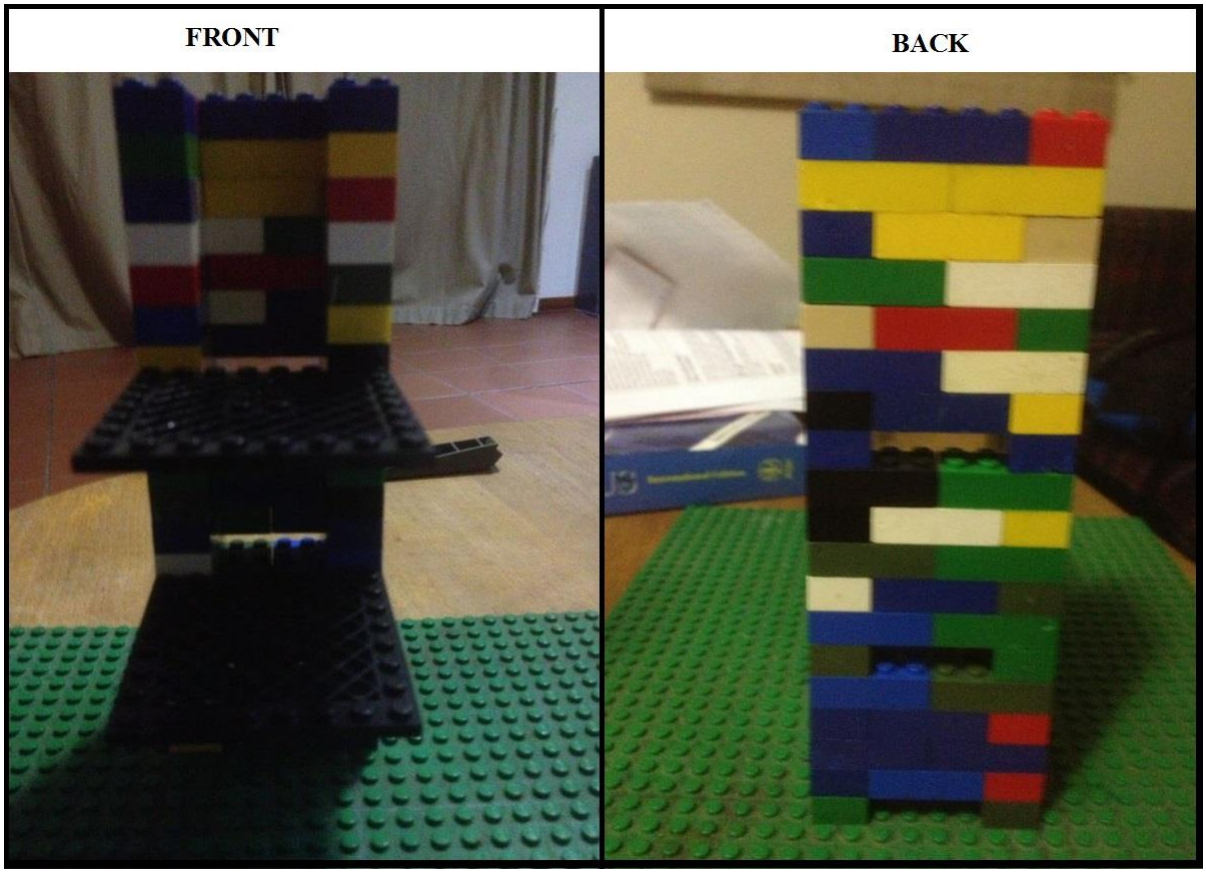


Figure 1B – Photograph of Initial Prototype Model

2. Final Prototype

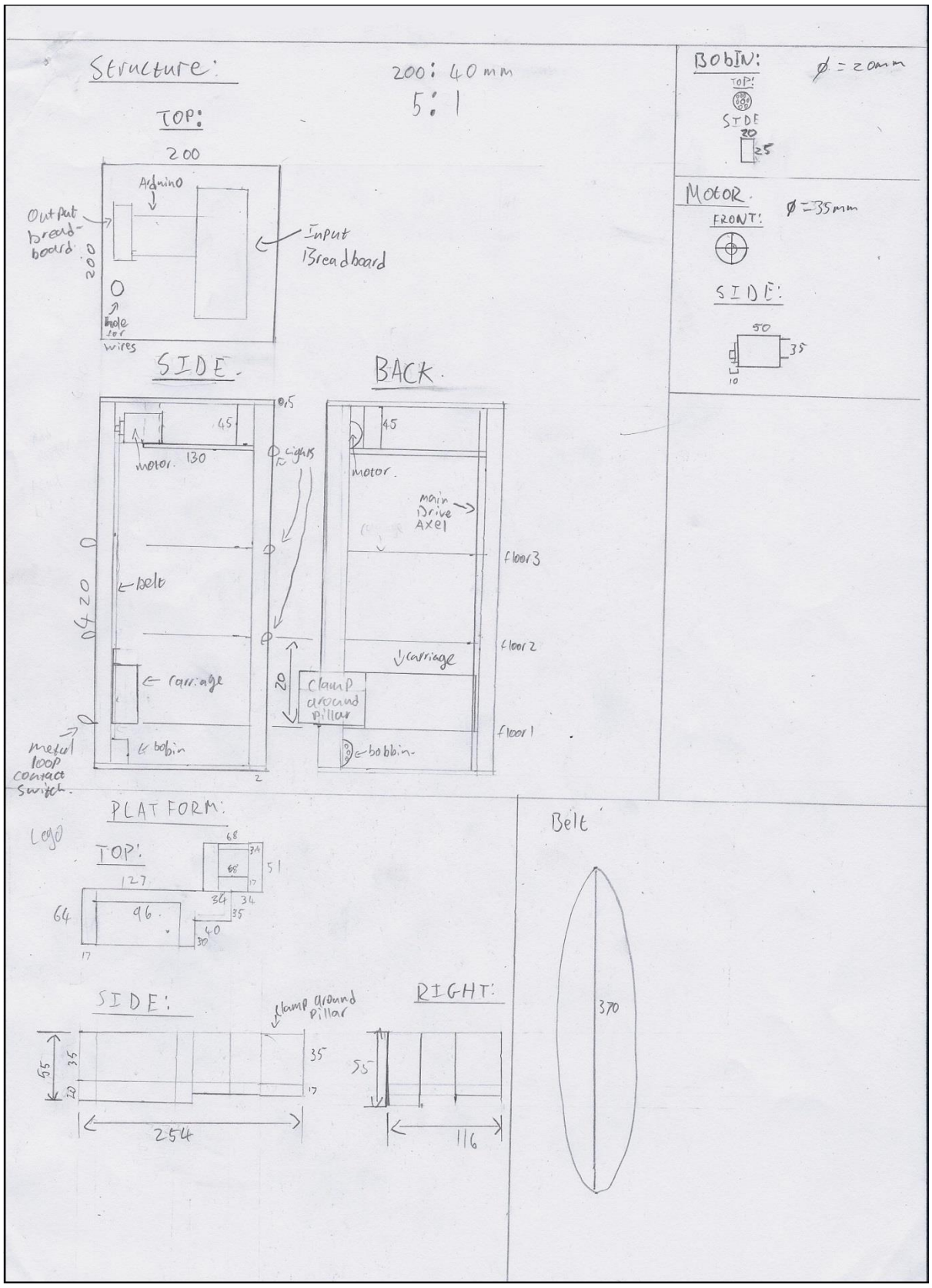


Figure 2A – Design of Final Prototype

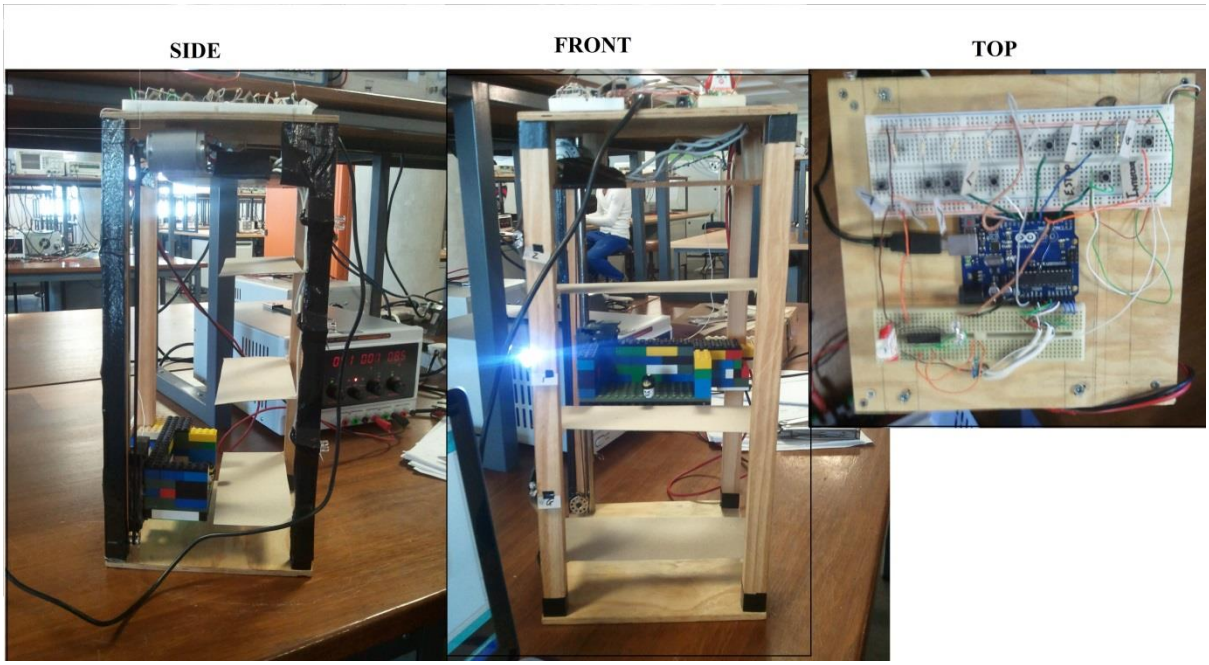


Figure 2B – Picture of Built Model