University of South-Eastern Norway
Institute of Science & Industrial Systems,
Kongsberg

NAGINI

Prototype documentation

by

Mehtab Singh Virk
Sameed Ahmed
Han Zhou

June 1, 2020

Abstract

This document outlines prototype documentation of a developed, robotic snake; Nagini. An accompanying systems engineering management plan (SEMP) and project management plan (PMP), can respectively be found in [1] and [2].

Table of contents

List of Figures

List of Tables

1 Stakeholder & System Requirements Definition

1.1 Mission Analysis

Per [3], the mission of project team during project-of-interest is "to construct a snake robot, that can move from arbitrary point A to point B on a horizontal plane". However, the mission is not limited to this statement.

See section 6.4.1 in [4]. The opportunity space arises from immediate stakeholder's need; any robotic snake that can move from an arbitrary point A to point B will per agreed definition suffice as a solution. This means that the solution space is characterized as infinitely huge, as any slight variant to a specific solution n_0 will give result to next solution n_1 and so forth.

In order to configure the mission statement to a particular, feasible solution, project team has decided to fulfill needed functionality with system-of-interest as an object follower robot. See Fig. 1 for reference. An object follower robot's purpose is to follow a detected or identified object to its best abilities. It is a common type of robot described by different design and implementation approaches in [5], [6] and [7]. In terms of a robotic snake, this operational function can be achieved in many ways. For system-of-interest description showing an overview of our implementation in terms of hardware- and software, see section 3 in [1].

Furthermore, the mission includes production of project deliverables as described in Table 1 in [1].

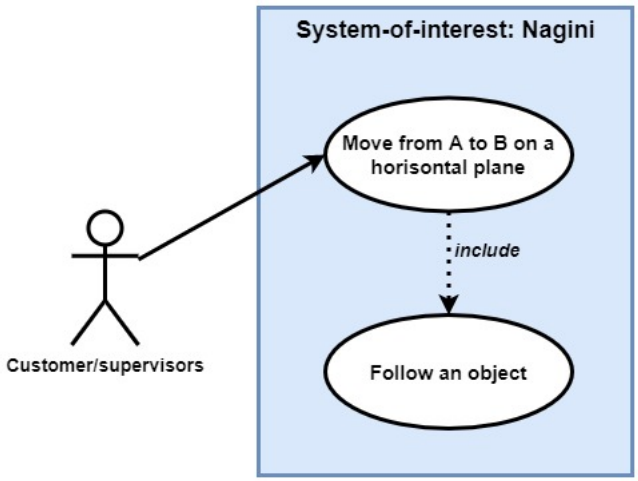


Figure 1: Use-case diagram showing high-level functionality of Nagini

1.2 Stakeholder Requirements

Table 1: Stakeholder Requirements Specification

SH ID	Req. ID	Description	Req. Type	Req. Priority
SH-001 Customers/ Supervisors	SH/R001	The system shall move from arbitrary point A to point B on a horizontal plane.	Functional	HIGH
SH-002 Project Team	SH/R002	The system shall follow an arbitrary object.	Functional	HIGH

See Table 1 describing stakeholder requirements for system-of-interest. The provided stakeholder requirement is SH/R001, while SH/R002 is a derived requirement by project team, introduced in section 1.1. Stakeholder requirement SH/R001 describes the functional capability needed by stakeholder SH-001: Supervisors. However, expressed need and thus stated requirement does not describe environmental, safety or otherwise conditions under which requirements must be met, leaving a set of unconstrained attributes to be set by project team. In order to develop a feasible prototype, project team has chosen underlying, simple conditions for the requirements; the environment in which system operation takes place must be dry, leveled, and safe. The safety aspect of system requirements is informally agreed upon as non-threatening, non-lethal and non-hazardous.

Moreover, as requirements SH/R001 and SH/R002 describe the functionality of a solutional system, the interface and interaction between user and system is undefined. Project team has taken into account human capabilities and skill limitations while designing user-system interface and interaction. For user documentation see section 8.

1.3 System Requirements

Table 2: System Requirements Specification

Stakeholder Requirement	Rationale	System Requirement	Description
SH/R001	Movement of system (point A to B)	S/R001	The system-of-interest shall move from point A to B.
SH/R002	Object detection and following	S/R002A	The system-of-interest shall detect an object within a specified range of distance.
		S/R002B	The system-of-interest shall follow an object that is within a specified range of distance.
		S/R002B1	The system-of-interest shall turn in order to follow an object that moves in two dimensions.

For reference, see section 6.4.3 in [8]. The set system requirements are described in Table 2. The stakeholder requirements given in section 1.2 is translated to system requirements. The difference between stakeholder and system requirements is that of high- and low level, problem domain and solution domain; stakeholder requirements are requirements that describe high-level functionality of a solutional system (being in the problem domain), while system requirements are derived, functional requirements for a selected solutional design. The transition of problem domain to solution domain happens when stakeholder requirements are translated to system requirements.

The system requirements can further be allocated to subsystems that constitute system-of-interest, i.e. specifically assigning functions to parts of the system that are required to perform them in order to fulfill system requirements.

2 Architecture Definition

Snake Robots have progressed substantially in the past few years. There is a lot of focus nowadays on enhancing the Robotic technology or Snake Robotic technology without human beings involvement in order to minimize any danger to human life which in effect requires new or improved technologies. This Chapter looks at Architectural definition for Snake Robot.

The following topics will be covered in this chapter:
• A Architectural view of how to build the first iteration of Snake Robot.
• About usage of Architectural sketches of Snake Robot.

See Appendix A for models of first prototype build.
From a back-end perspective, the main focus in the Architecture process was on project scalability, thus, a function has been designed in the Architecture of the Snake Robot head in model 9a of App. A which allows the administrator to quickly add new functionality to the function.

The Architecture Definition contains the deliverable for the core architectural artifacts created during a project and for important related information. The Architecture Definition spans all architecture domains and also examines all relevant states of the architecture. The Architecture Definition provides a qualitative view of the solution and aims to communicate the intent of the architects.

2.1 Snake head

See model 9 and 9a for the architectural sketches of the snake head with the measurements of different component placed at the front part of the snake which is called Snake head. Calculations of Arduino, DC motor drive, Battery, wheels, DC wheel and Servo motor are given and can be seen from both architectural models 9 and 9a. Height, Width and length of the Snake head can also be seen in the figure 2. Arduino, DC motor Drive, battery and Servo motor are placed inside the head of the snake while wheels and DC motor are outside and at the bottom in the compartment unders the snake head. There is a place at the face of the snake robot for Sensor to detect an object at front of the Snake.

2.2 Snake robot overall architecture

Model 10 in App. A shows the overall architecture of the Snake Robot. Four links have been used to create the back of the Snake robot. Distance between each link can be seen from the model. Each link has servo motor installed inside it for the rotation of the link in order for the snake to turn right or left.

2.2.1 Top-view

Model 3 in App. A shows the architectural sketch of the overall snake robot body from the top. At the front there is a head of snake having all the important components after that, there are multiple links connected by each other with joint. Two different type of joints have been used in the first and later iterations. In first iteration Legos made joint has been used while in the later iterations, custom designed universal joints have been used. Wires between different links and head are gone through the bottom of the snake body with clips connected in order to hold them firmly.

2.3 Link: C-bracket

See model 1, 2 and 2a in App. A for design and measurements of a link, i.e. C-bracket and servo motor. These models show the architectural design of links. Height, weight and depth has been precisely calculated and shown in these models. In model 1, the architectural sketch shows how the servo motor is fitted inside the c-bracket. Model 2 shows how different links has been designed to connect the C-Brackets with each other to complete the back of the snake. In model 2a the architectural sketch shows detailed dimensions of the C-bracket and Servo motor.

2.4 Wheel- and shaft placement

Model 5 in App. A shows the architectural sketch of the snake robot wheel and shaft connection which are essential for the motion of the snake. When the servo motor rotates 90 degrees linear motion has been created, while turning snake left or right it hits the wheels thus creates a problem which has been solved in the next architectural sketch given in model 9.

Model 11 and 11a in App. A shows the architectural diagram of the C-bracket view from the bottom and side respectively. Details for all the dimensions of non elecctronic-components connected with the C-bracket are also given. Model 11 shows lego technic beams placement underneath links, while model 11a offers a side view of same profile. In addition, model 11a contains dimensional sketch of the used wheels and their thought placement.

2.5 Joint: Ball joint (deprecated)

Model 7 in App. A shows the architectural diagram of the new type of ball joint to connect the different links of Snake using C-bracket which has been deprecated in favor of a universal joint adopted in the second development iteration of hardware.

2.6 Movement subsystem (beneath head)

Model 13 in App. A shows the architectural diagram of the movement subsystem with DC motor and wheels which are connected underneath the head. Different dimensions are mentioned in the sketch shown above. This architectural design of Movement subsystem has been adopted in the later development iterations of the snake robot as well as in the first iteration.

2.6.1 DC Motor

Model 13a in App. A shows the architectural diagram of the DC motor which is connected with wheel and lego technic beams to form a part of the movement subsystem. All the dimensions for DC motor have been precisely written in the sketch shown above. This design of DC motor has been adopted in the first development iteration of the snake robot as well as in the second iteration. While model 13d shows the base of the DC motor with the wheel on the side. The compartment in which DC motor has been placed under snake head is also shown in model 13d.

Model 13d.1 in App. A shows the architectural diagram of the Snake robot from the bottom. All the dimensions of Snake robot have been precisely written in the sketch shown in model 13e.1 and 13e.2 respectively. This design of Snake robot has been adopted in the first development iteration of the snake robot as well as in the second iteration.

2.7 Detection subsystem

Model 4 in App. A shows the architectural diagram of the detection subsystem of the Snake robot. Architecture of the Ultrasonic sensor and IR obstacle avoidance sensor has been given in model 4. This design of Snake robot has been adopted in the first development iteration of the snake robot.

2.8 Component measurements

Model 8a in App. A shows the architectural diagram for the measurements of different components of the Snake robot. Architecture diagram for the dimensions of the Ultrasonic sensor, DC motor, Motor driver, Arduino and battery has been given in the figure 20 above. This design of Snake robot has been adopted in the first development iteration of the snake robot as well as in the second iteration except for the sensor.

2.9 Placement of components in head

Model 14a in App. A shows the architectural diagram for the placement of components on the Snake robot head. Architecture diagram shown above have dimensions for the Servo motor, DC Motor driver, Arduino and battery. This architectural design of Snake robot head also shows how all the component are placed in the snake head with width, height and length of every component. This design has been adopted in the first development iteration of the snake robot as well as in the second iteration.

Model 14b shows the architectural diagram of the Snake head with the Snake robot movement subsystem at the bottom of the head. Architecture diagram shown above have dimensions for the DC motor, Shaft and wheel. This design of Snake robot head also shows how all the component are placed in the snake head at the top above the detection subsystem. This architectural design has been adopted in the first development iteration of the snake robot as well as in the second iteration. While last model 8 shows the different architectural design iterations of the first snake robot head.

3 System Analysis

In System Analysis the main purpose is to study a system which we have developed or parts of it. By studying the system, we can ensure that every component is work according to their required functionality and working in an effective manner. What the system should do is a part of a system analysis.

In the initial phase of system analysis when it comes to analyze the data and components, there has been developed a trail test for the people who want to check the performance and analyze it according to the functionality stated in the project description which is correct or not. First thing which was needed was a platform to visualize the behavior which selected to be the Zoom meeting as campus is closed due to disease control restriction which is best in its domain in order for other colleague to check the functionality online by giving their requirement to the person controlling the Snake robot.

Due to a fixed timescale and having requirements which might change during development and Design section, this project has been developed using the Agile model. The hardware and software has been developed in incremental stages, rapid cycles having each release building on previous functionality and research.

The Agile methodology is ideal for building software and hardware where requirements are subject to change during the development life cycle. The methods are based on the Agile Manifesto which was published by a group of software developers in 2001, see [9].

See the connectivity diagram shown in Appx. C. The diagram is the analysis of the all the component which has been placed in the Snake head. All the connections are also visible in the diagram. Function description has also been written besides every component. These are all components that has been used in the Snake robot during the final development iteration.

Figure 2: DC Motor

Fig. 2 shows DC motor which is connected with wheel and C-bracket. All the dimensions of DC motor have been precisely written in section 2. The rotation of DC motor results in the motion of the snake Robot. This DC Motor has been adopted in the first development iteration of the snake robot as well as in the second iteration. While model 13d in App. A shows the base of the DC motor with the wheel on the side. The compartment in which DC motor has been placed under snake head is also shown in model 13d and Fig.19 in App. A and section 2 respectively.

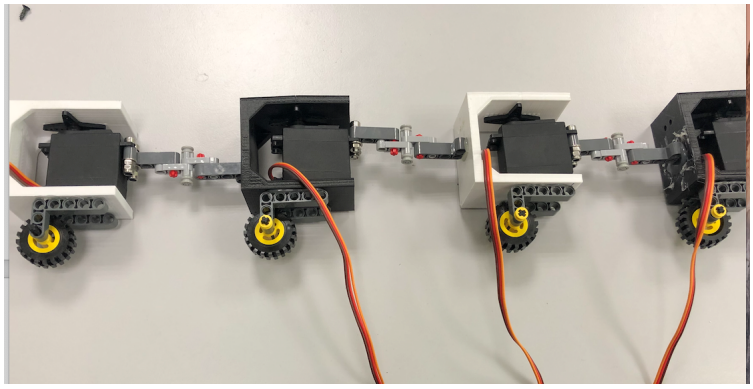


Figure 3: Links with joints v0.1 made of lego

Fig. 3 shows the actual look of the back of the Snake Robot. Four links have been used to make the back of the Snake. Each link has servo motor installed inside it for the rotation of the link in order for the snake to turn right or left. The joint above in Fig. 3 between each link has been adopted in the first development iteration of the snake robot but discard in the second iteration. The new type of universal joint which has been used in the second iteration can be seen in Fig. 16. While Fig. 4 is the actual photo of the DC motor driver which has been used in the first development iteration as well as in the second iteration.

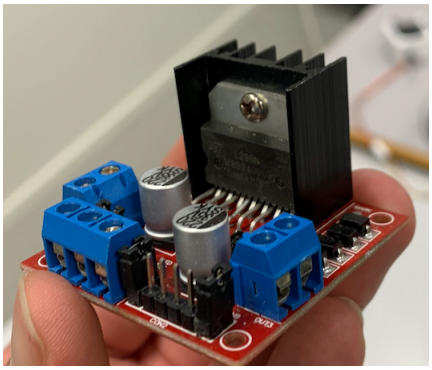


Figure 4: DC Motor Driver

4 Design Definition

Snake Robots that exist in the world are of many types. Small Snake Robot which have the speed between 1 to 20 km/h. Also, there are micro Snake Robot which are as small as 15cm with a mass of as low as 80g. There are Snake Robot that are Ultra small, less than 15cm and are lightweight having a mass as low as 20g. Performance of Snake robot depends on both effectiveness and highly responsive motor control. The good design and control of Snake Robot is getting even better with each passing day. Operations of Snake Robot have increased 100 times more than the operations used to be few years back. It is important to design each component of Snake robot with precision otherwise it can result in a failure and waste of resources.

The list of the components for the first build of Snake Robot is given below:

- Arduino Mega
- 12V Battery
- DC Motor (12VDC, 300RPM)
- Motor Driver
- Mini Lidar
- C-Brackets (Link) (4 in total)
- Joint (4 in total)
- Snake Head
- Servo Motors (5 in Total)
- Wheels (5 in Total)

Figure 5: Face Design

Fig. 5 shows the final design of the snake face with the measurements given in the architecture section. Height, Width and length of the Snake head can also be seen in model 9a in App. A. The sensor has been placed at the bottom area of the snake robot face and is used to detect an object in front of the Snake.

Figure 6: Initial laser cut design of backplate of C-bracket

Fig. 6 shows the initial design of the front side of the C-Bracket which has been made with laser cut printer with the measurements given in section 2.

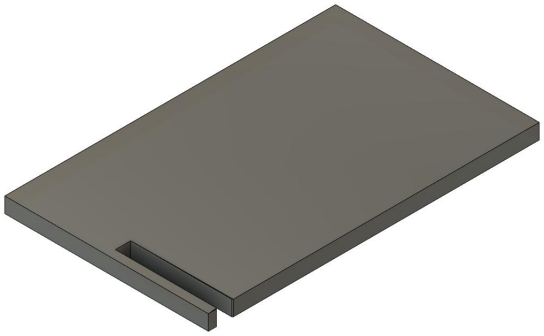


Figure 7: Initial laser cut design of bottom plate of C-bracket

Fig. 7 shows the inital design of bottom side of the C-Bracket with the measurements given in section 2. It has been made with a laser cutter.

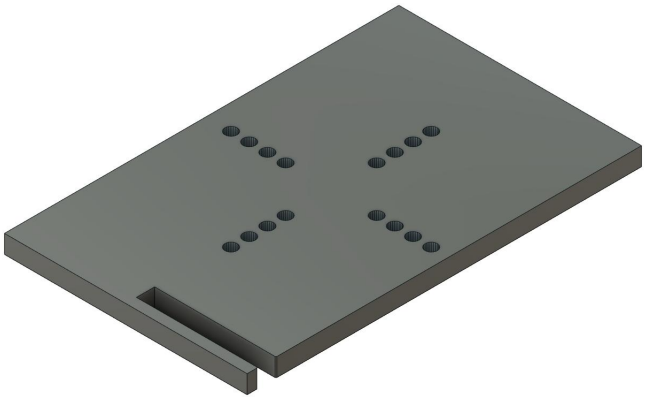


Figure 8: Initial laser cut design of top plate of C-bracket

Fig. 8 shows the initial design of the top side of the C-Bracket with the measurements given in section 2. It has been made with laser cutter as well.

Figure 9: Initial laser cut design of assembled C-bracket

Fig. 9 shows the initial design of the C-Bracket with all its sides assembled together. This design has been made with laser cutter. This design of C Bracket was adopted initially in the first development iteration of the snake robot, but was discarded after 3D printing possibilities were known.

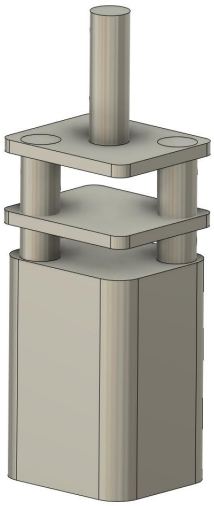
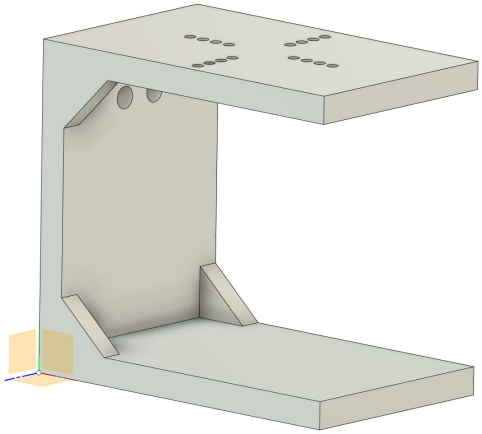
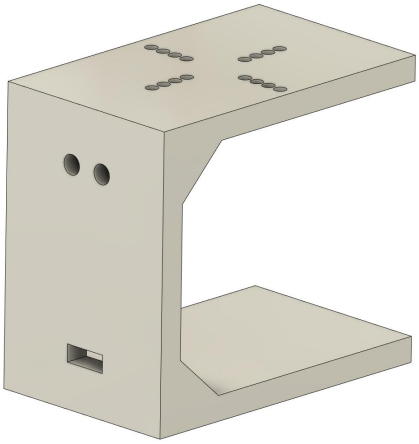


Figure 10: Design of DC motor

Fig. 10 shows the 3D design of the DC motor which is connected with wheel and C-bracket. All the dimensions of DC motor have been precisely written in the architectural section. This design of DC motor has been adopted in the first development iteration of the snake robot as well as in the second iteration. While model 13d in App. A shows the base of the DC motor with the wheel on the side. The compartment in which DC motor has been placed under snake head is also shown in same model.



11.1: Link v0.1



11.2: Final design of link (v0.2) with passthrough for servo motor cable

Figure 11: Two different versions of links illustrated.

Fig. 11.2 shows the final design of the C-Bracket. This design has been made with 3D printer. This design of C Bracket has been adopted in the first and later development iterations of the snake robot.

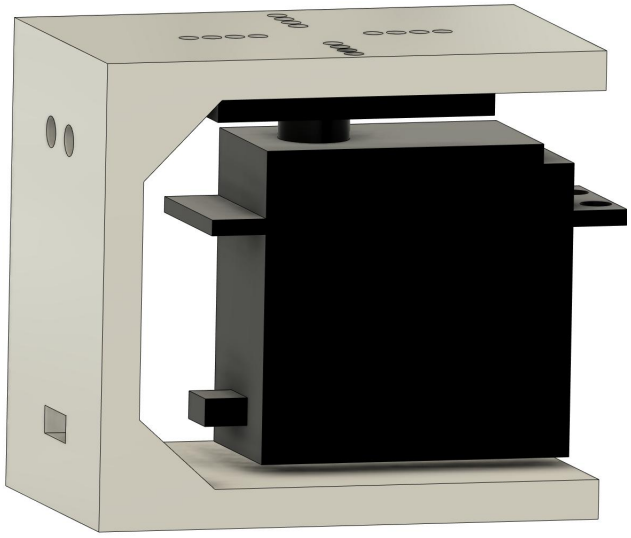


Figure 12: Design of C-bracket with mounted servo motor

Fig. 12 shows the final design of the C-Bracket with servo motor installed in it. This design of C Bracket has been adopted in the second development iteration of the snake robot. The Servo motor has been fixed with four screws from the top side of the C-Bracket.

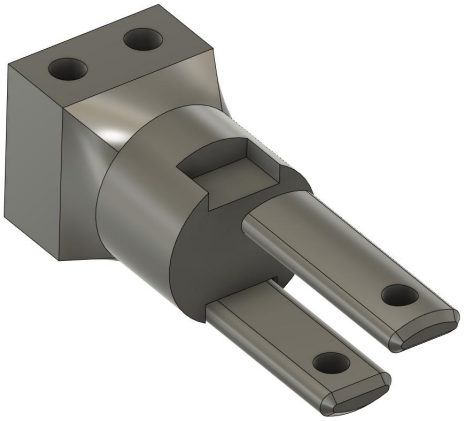
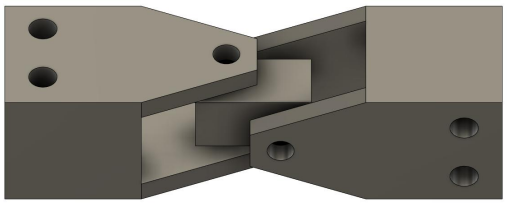
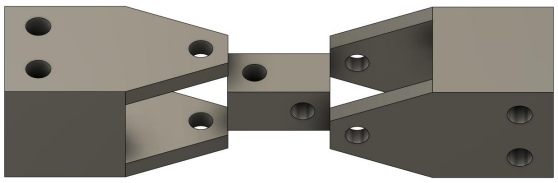


Figure 13: Design of universal joint v0.3

Fig. 13 shows design v0.3 of the universal joint which connects with the C-Bracket to act as a part of joint between two C-bracket. This design of universal joint was developed to test if the arms could be glued onto holding places and still function. The joint used in the first iteration has been shown in the system analysis chapter in the figure 45.



14.1: Universal joint v0.4 ready to be screwed to central block



14.2: Universal joint v0.4

Figure 14: Final design of universal joint (v0.4)

Fig. 14 shows the design of the final universal joint. This design of joint joint has been adopted in the second development iteration of the snake robot.

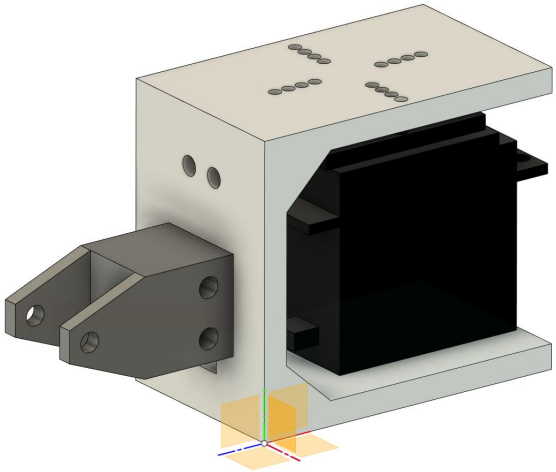


Figure 15: Link with part of joint attached

Fig. 15 shows the design of the C-Bracket with part of universal joint attached on one side and servo motor installed on the other side of it. This design of C Bracket has been adopted in the final development iteration of the snake robot.

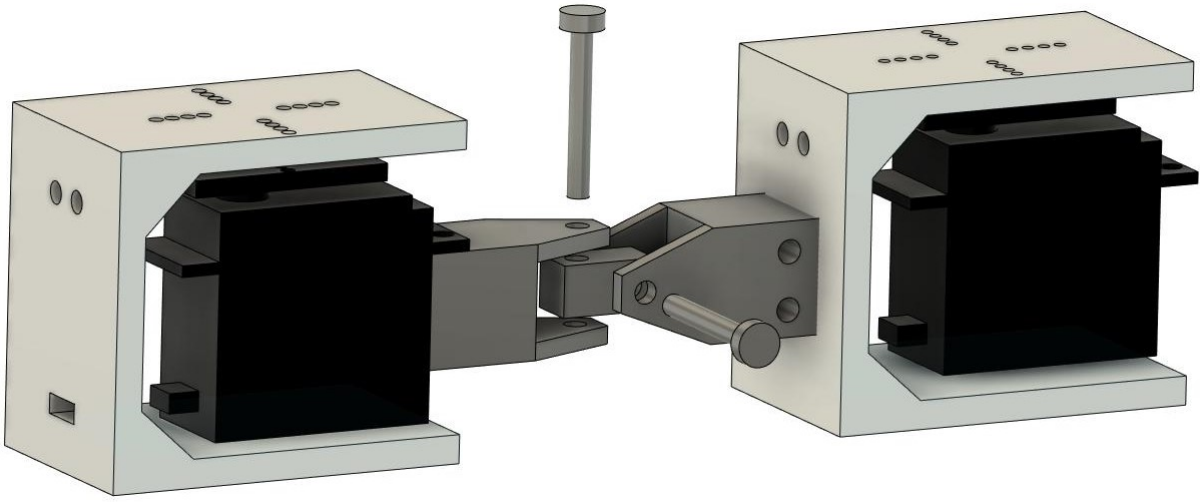


Figure 16: Final design of link and joint

Fig. 16 shows final design of link and universal joint integrated. This design has been used in the final development iteration of the snake robot.

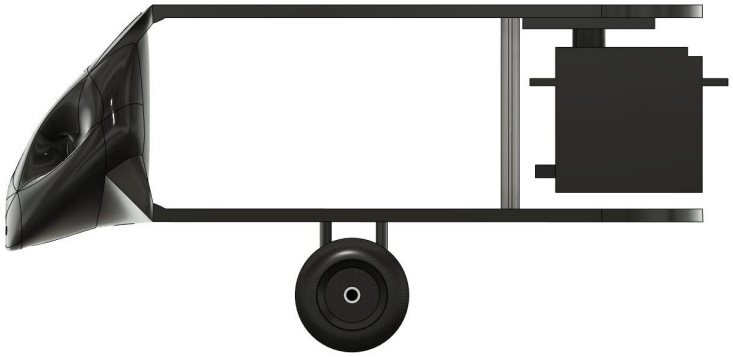


Figure 17: Side-profile of snake head v0.2

Fig. 17 shows the design with the side-view of the Snake head v0.2 connected with the servo motor on the back and face. This design of Snake robot has been used in the second development iteration.

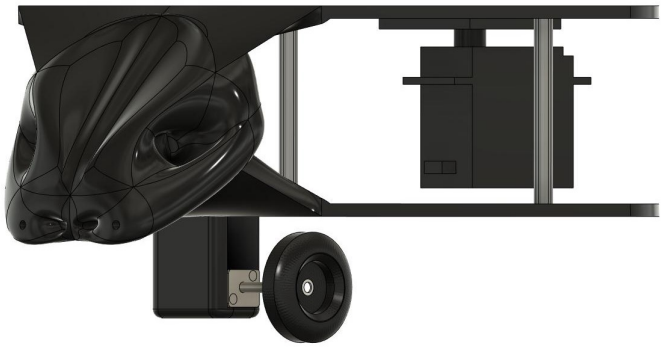


Figure 18: Snake head v0.2 from an angle

Fig. 18 shows the design with the tilted view of the Snake head connected with the servo motor on the back. This design of Snake robot has been used in the second development iteration. The Wheel and DC-Motor is also visible in this design. DC motor has been installed in the compartment at the bottom under the snake head.

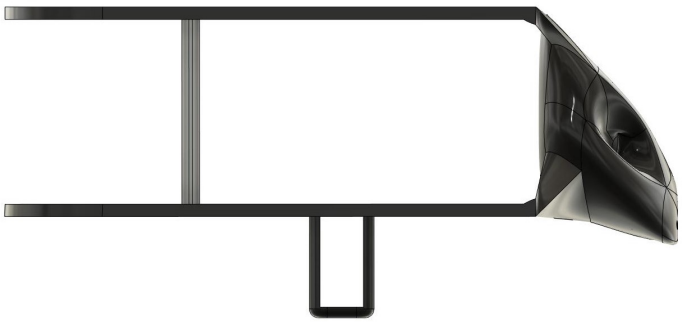


Figure 19: Other side-profile of snake head v0.2

Fig. 19 shows the design with the side view of the Snake head v0.2 with the DC-Motor compartment shown at the bottom under the snake head. This design of Snake robot has been used in the second development iteration.

Figure 20: Snake head v0.1

Fig. 20 shows the design of the Snake head v0.1 with face and DC compartment. This design of Snake robot has been used in the first development iteration but has been discarded at the later stage of the project due to failure of right/left movement of the snake with Servo motor stuck because of improper positioning.

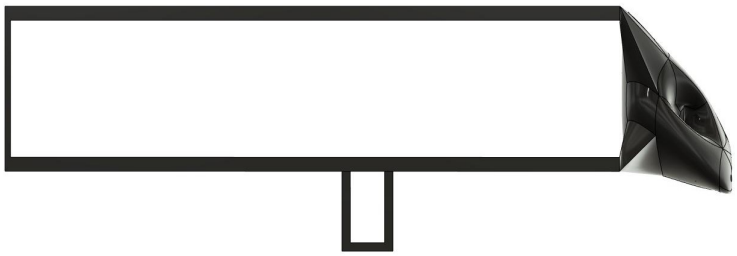


Figure 21: Side-profile of snake head v0.1

5 Implementation

The implementation of the Snake Robot is a realization of a technical specification and the execution of plan which is finalized in the design and architectural section above.

In the initial stage of implementation we have performed two important tasks: First got access to the hardware we needed to assemble the Snake robot head and back of the Snake Robot, list of all the component used in order to assemble the front and back of the Robot is given in the design chapter of this document. Then we gathered all the requirement for the Software development of the Snake Robot. Next step in the process used was what parameters and attributes are there and how to use them to get our Snake hardware and software working together properly and according to the functionality set at the start of project. Afterwards, we took trail test of the Snake robot to check linear motion of the robot starting from initial point A to destination point B. In order to get the Snake to move from point A to B we used software/hardware configuration set to rotate the DC motor with the Shaft for the snake head to move forward in order to produce motion.

The software used for the snake Robot is Arduino C++. Arduino is an open-source hardware and software development kit, which is used for hardware/software development project to design microcontroller for building digital circuits.

In the Second iteration of implementation we have improved the existing solution and added the functionality of Snake Robot moving towards left or right. Also, Object collision avoidance mechanism used to avoid collision with object which are in front of the Snake robot. Different type of object is detected using the Mini lidar sensor placed at the head and front of the Snake Robot alternative to snake eye. Distance set for the snake to stop if there is an object in front is 30cm which can be change if requirement can change. Mini lidar sensor use light to calculate distance and calculation checked multiple time to verify the accuracy and precision of the calculations. Same step is used here in the process which was used previously in the first iteration to check what parameters and attributes are there and how to use them to get our Snake hardware and software working together properly and according to the functionality set at the start of second iteration. Afterwards, we took trail test of the second iteration of the Snake robot to check linear motion involving moving towards right or left and also verify object detection of the robot in front with starting from point A to destination point B. In order to get the Snake to move from point A to B we used software/hardware configuration set to rotate the DC motor with the Shaft for the snake head to move forward in order to produce motion.

Next the Property tree gives us all parameters for Mini lidar Sensor place at the head of the Snake robot. Whatever is going behind the Lidar Sensor can be seen from the Property Tree values. If we change the snake robot position, the values will also be changed in the property tree. If the snake is moving forward or turning left or right then we will receive different values at every next instant. We can view that change motion of the snake robot can also change the value in the mini lidar sensor. There are features like acceleration, speed and angle of rotation can change on every instant when snake is moving. After receiving data and information about different parameters and after the analysis from the Property tree, accuracy of trail test was verified and functionality set the start of the second iteration checked

In the end of the implementation phase we have concluded that Snake robot are very important for different kind of application civil as well as military applications. Snake robot have many advantages one and most important of them is that, they can used to perform different task which are difficult or in some situation impossible for the human to perform. Another advantage of Snake Robot is they save a lot of time to perform some task as compared to human being doing it and also Snake robot can perform it in more effectivity and in best possible way. With the advancement in their batteries, applications, reducing cost, small size and power made people to attract more towards them. Their applications in our life is increasing day-by day. The time is soon when everyone going to have their own Snake robot for their personal needs.

6 Maintenance

The maintenance will follow the standard ISO/IEC/IEEE 12207:2017 [4] and ISO/IEC/IEEE 15288:2015 [8]. The maintenance procedure will include the stakeholder management plan and the technical plan.

6.1 Stakeholder Management Plan

Maintenance factor to the human factor is a normally little considered workload compare to the system engineering effort. In our project, we tried to make some effort to take into account as mush as possible to this function point. The questions can be raised are, is the service plan good enough? How shall we ensure an effective maintenance deployment? Is the maintenance plan as expected? Is the maintenance activities as required? To fulfill these questions in a professional way, we adapted some techniques from the Scrum Master approach. Scrum is rather a framework than a methodology, it is widely adapted in the software engineering world, since 1995, developed by Jeff Sutherland and Ken Schwaber in Austin [10]. See table 3 for a maintenance tasks plan toward the internal and external stakeholders.

Table 3: Maintenance plan

Stakeholder	To do tasks	Request	In progress	Done
Project sponsor	Ensuring the understanding of maintenance product domain			
	Identified approach			
	Ensuring the understanding of the maintenance plan			
	Facilitate events as necessary			
Maintenance Team	Coaching the team in self-organized and cross-functionality			
	High valued productivity			
	Identifing and removing impediments from the event progress			
Customer	Leading the maintenance events in Scrum adoption			
	Help planning the Scrum implementations			
	Ensuring the effectiveness of the Scrum approach			

6.2 Technical plan

Then maintenance procedure in technical part will include the failure diagnosis, the components, and assembly instructions. The software application shall be archived, shall have backup for recovery possibility and be managed in the information management plan.

7 Software & Hardware Integration

Our product is a robot system which mimic the behavour and movement of a snake. The product is consisted of three parts: the mechanical frame in terms of a snake form, the Arduino system and the software. The integration is generally conducted by the ISO/IEC/IEEE 12207:2017 [4], that is referred in the ISO/IEC/IEEE 15289:2019 [11]. The integration of our prototype will thus in three phases, the mechanical assembly phase, the Arduino system assembly phase and the software integration phase.

7.1 Mechanical assembly of first build

7.1.1 The Mechanical Parts

The mechanical parts are the links, joints, wheels and cables which composed the snake form. In Fig. 22 is one set of wheels that shall be placed under one link.

Figure 22: One set of wheels

7.1.2 Mechanical Assembly Structure

The mechanical assembly structure shall be delivered together with the prototype

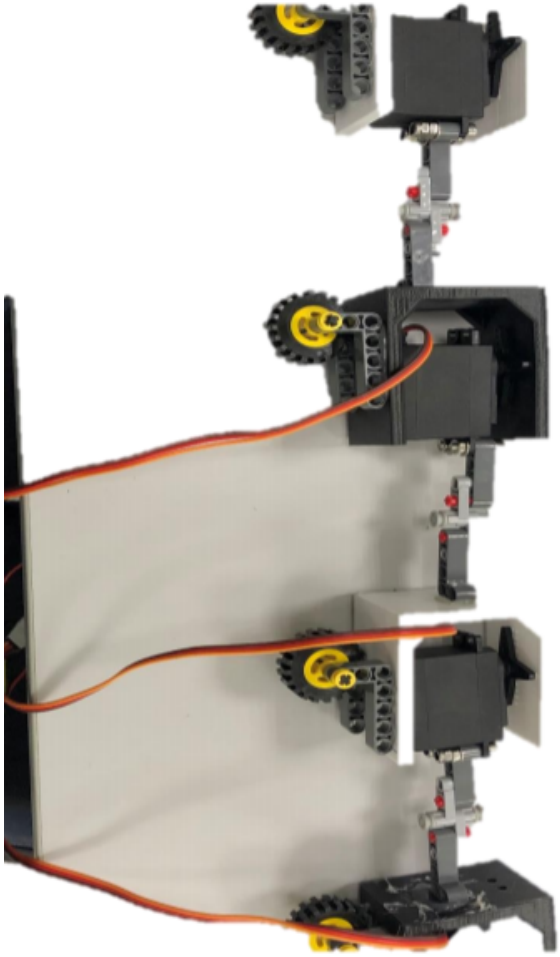


Figure 23: The first build of snake body

7.1.3 The Mechanical Assembly Process

The loss mechanical parts can be assembled as one build with Lego parts. The links shall be connected by the joints which are also Lego parts.

7.2 Arduino system assembly of first build

7.2.1 The Arduino Parts

The essential Arduino parts are the Arduino mega board, the DC motor, the DC motor driver, the external power supply, the servo motor.

7.2.2 The Arduino Assembly Structure

The main Arduino assembly structure is in Fig. 24, which is the prototype brain structure. There is the DC motor, one DC motor driver and the external battery to ensure the accurate voltage supply to the DC motor. The battery is 9V. In the middle is the mother board, the Arduino board. The sensor used in the prototype is for the distance detecting and measuring. The sensor type can be the mini-Lidar or the ultrasonic HC-SRo4. The software application will be different for different type of sensor. Below the battery and the mother board is the servo motor that is used to control the rotation of the links.

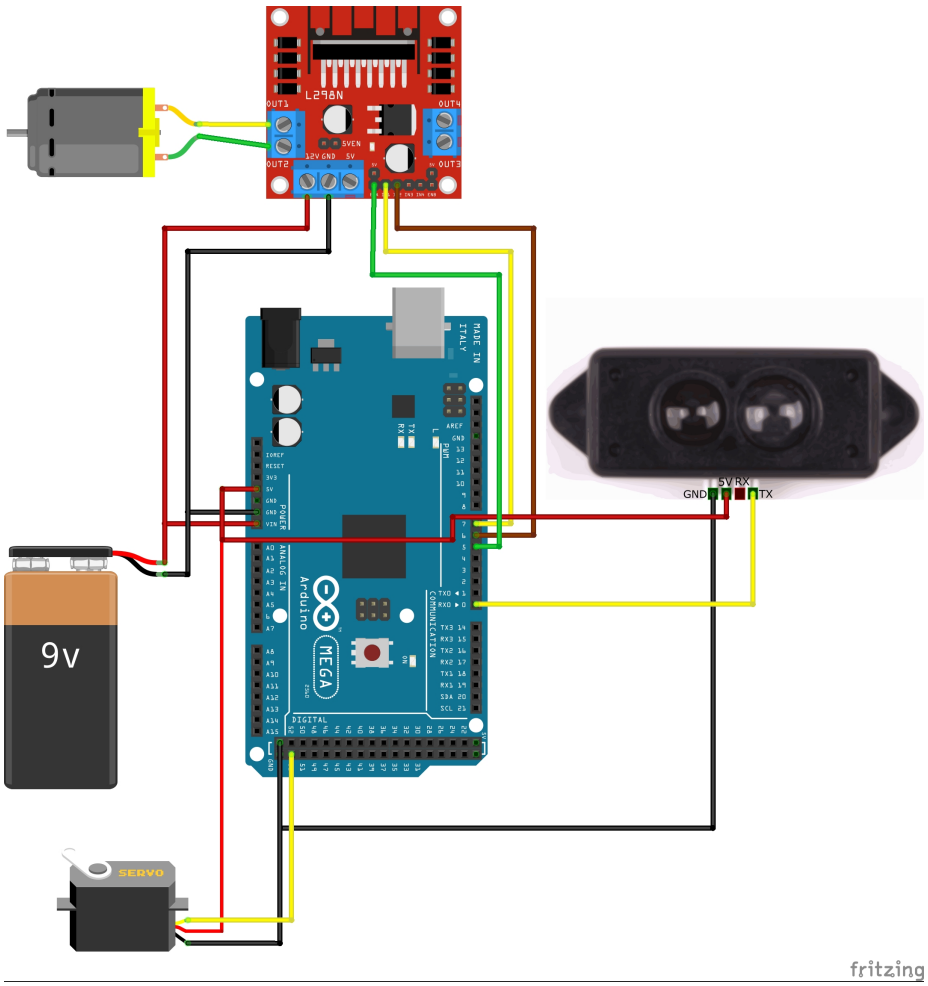


Figure 24: Main structure

However, if the LP298N in the figure is not available, there is another type of LP298N. The connection between the DC motor and the DC motor driver is the same, of which OUT1 and OUT2 on the board are connected to the DC motor. The power supply connection is as shown in Fig. 25, with the black cable to the GND and the red cable to the positive point.

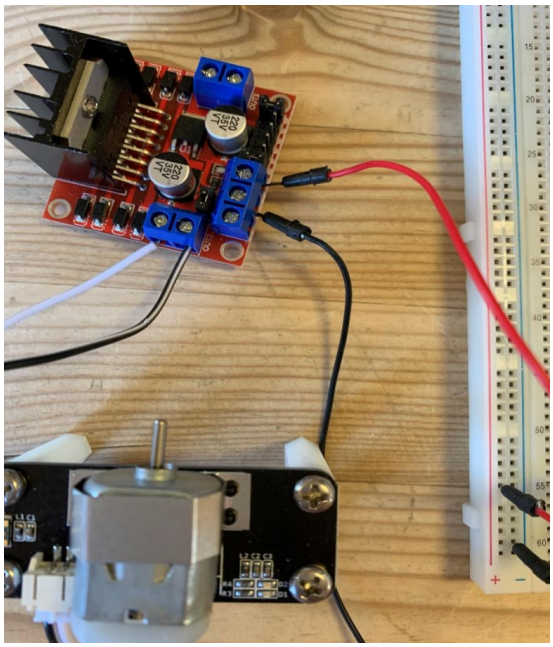


Figure 25: Alternative structure with DC motor

Furthermore there is an extended version of our prototype, which will be equipped with IR remote control, a led display and up to 4 servo motors (later build).

7.2.3 The Arduino Assembly process

The start the Arduino assembly process, it is necessary to have acquainted about the basic layout in a bread board, which will connect to the Arduino board for the extend system parts. As shown in Fig. 26, the red line on the left side of the bread board is the positive points for the voltage. The connectivity of the positive voltage on the bread board is along the red line, through out the bread board edge. The negative voltage connection is along the blue line. The horizontal connect points are connected in the bread board in a row basis. There is no electricity connections between the rows.

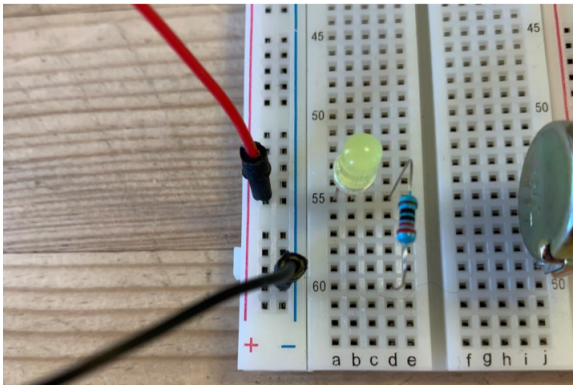


Figure 26: Bread board

Then we can start to connect the prototype system with the Arduino assembly structure showed in section 7.2. The detailed pin configuration can follow the connection table 4.

Table 4: Prototype pin configuration

Component	Pin Configuration
DC motor	OUT1 and OUT2 (as in connected to motor driver)
DC motor driver	5V, GND, EN_A(pin 5), IN1(pin 7), IN2(pin 6)
Mini Lidar	GND, 5V, TX(pin RX0)
9V Battery	V_in, GND
Led Display	5V, GND, SDA(pin 20), SCL(pin 21)
IR receiver	3.3V, GND, receiving pin (pin 22)
Servo motor #1	5V, GND, signal (pin 53)
Servo motor #2	5V, GND, signal (pin 51)
Servo motor #3	5V, GND, signal (pin 49)
Servo motor #4	5V, GND, signal (pin 47)

Then we shall configure the Arduino system with the computer, so that the Arduino board can be loaded with the software developed for the software. First we shall start the Arduino IDE program in the computer, then use the USB cable to connect the Arduino board to the computer. Please ensure to select the correct Arduino board in the program. When the connection is successful, the led on the Arduino board will turn on as shown in Fig. 27. the program will display the port be connected with the COM number.

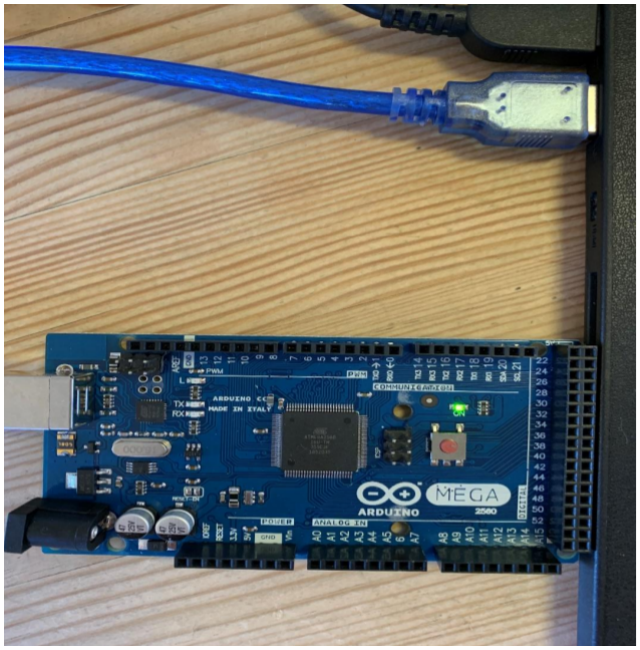


Figure 27: Arduino configuration

7.3 Software Integration

When the mechanical parts and the Arduino system have been assembled and integrated, then we can work on with the software integration. First we shall connect the Arduino board with the computer, shall be sure the Arduino communicate with the computer. For our prototype, its is the Arduino Mega with Atmega2560 processor. This can be checked in the Verktøy down window. As you can see in Fig. 28, the choice of "kort" is mega 2560, which is correct.

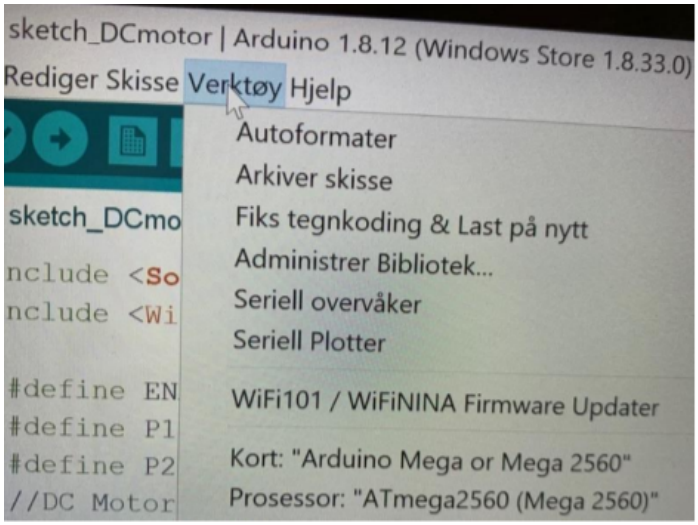


Figure 28: choose "Kort" Arduino Mega/Mega 2560

Then we can run the software by verify the code. This is to ensure the application code is ready to operate the prototype robot. When the verification is done without error, you will have a message similar to Fig. 29.

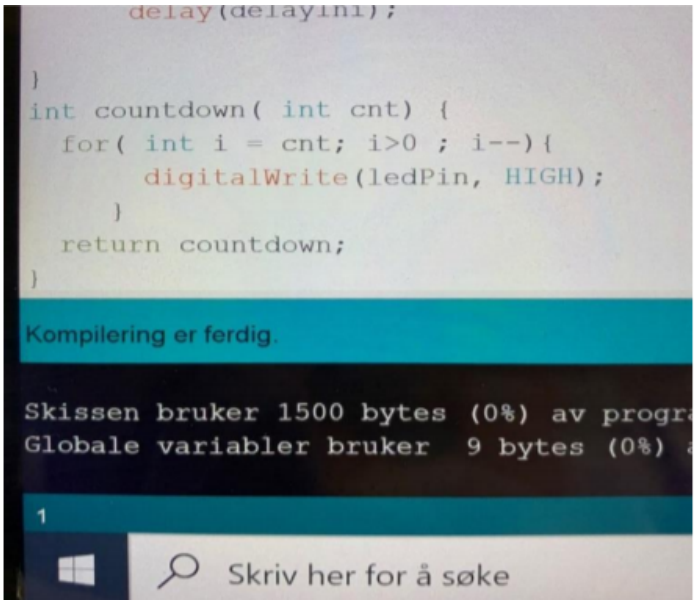


Figure 29: a success compilering

Next it is recommended to check the system integration is successful in a dummy. As shown in Fig. 30, the orange led is turned on to indicate that the application download to the Arduino board is

successful. Now it is ready to let the Snake Robot in action.

Figure 30: a success integration

8 User Documentation

This part of the documentation describes user related instructions on how to build and operate the latest version of Nagini. All components, parts and tools can be found in Appendix B in [1].

8.1 HowTo: Build Your Own Nagini(latest build)

1. 3D print links (5x), compartment (1x), joints (6x), central bricks (3x) and face (1x).
2. Glue a leg of the joint on 3 of the printed links. Keep in mind that the joint goes a little above the hole, and approx. 1cm from each side of the link. See Fig. 31 for reference. It is okay if the part gets glued a little misaligned.

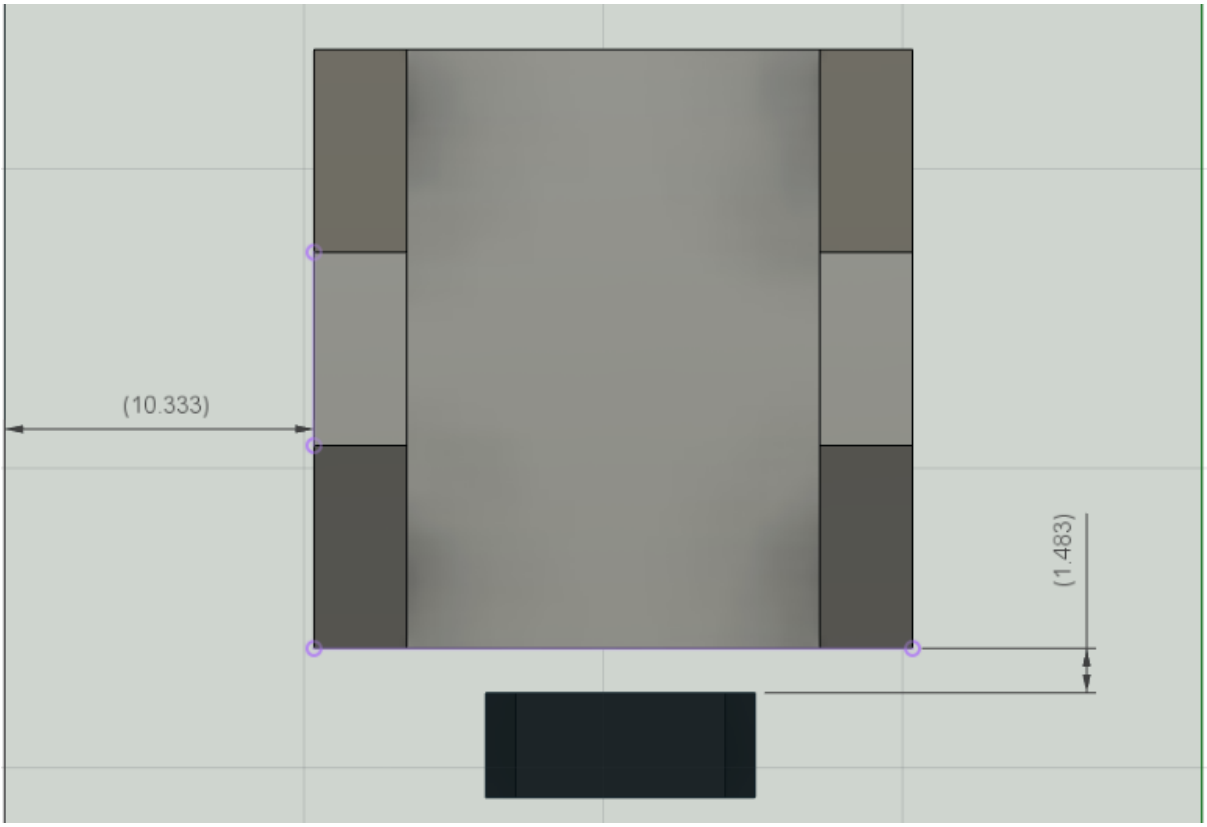


Figure 31: Joint positioning on link. All units in mm.

3. Glue the remaining legs (of printed joints) to the servo motors. These parts of the joint should be glued right beneath the screw-holding tab of the servo motors, as depicted in Fig. 32.

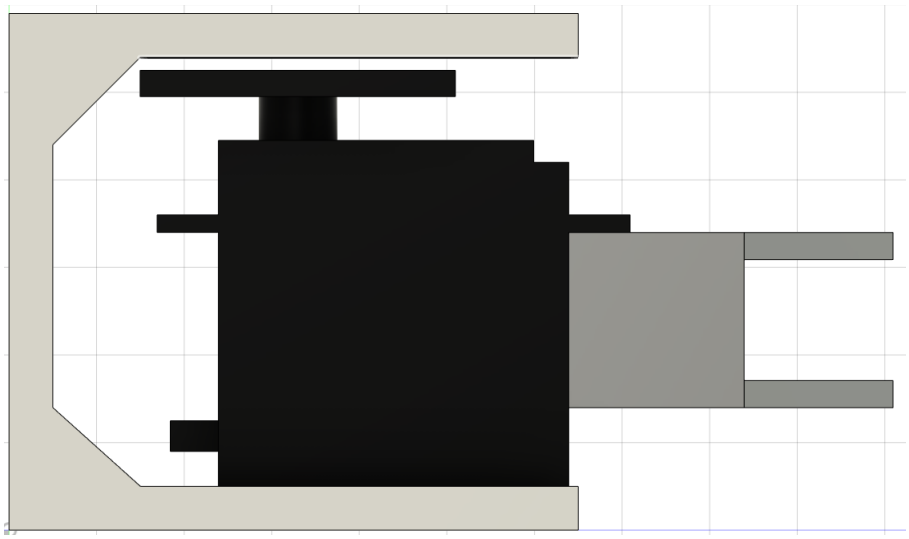
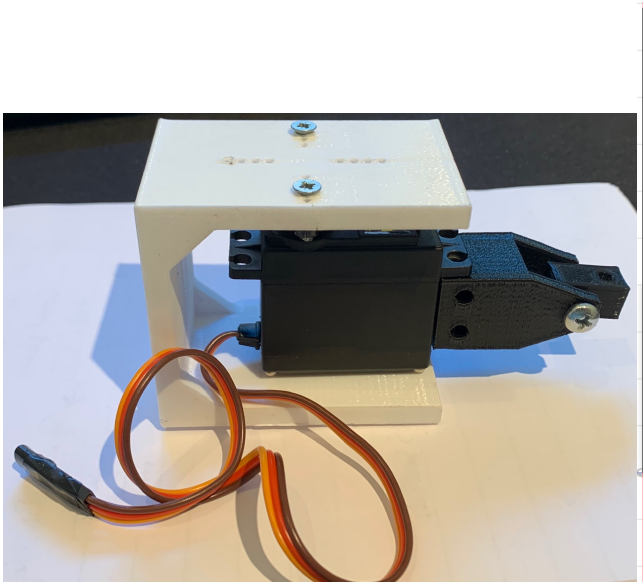
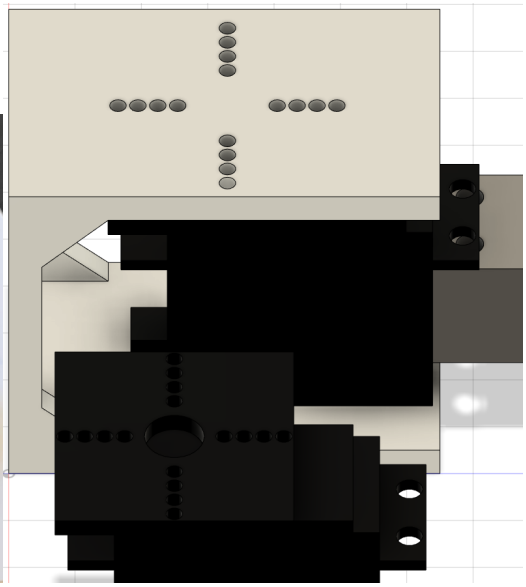


Figure 32: Joint position on servo motor.

4. Align the holes on servo motor head (see Fig. 33.2) with the holes on top of link. Hold head of servo motor tightly against the roof of the link and screw the servo motors to their links with M3 flathead screws, as shown in Fig. 33.1. Note that there should be a little space from bottom of servo motor to link. This space is needed so that the servo motor can rotate without scraping bottom of the link.



33.1: Servo motor screwed to link



33.2: Holes in servo motor have to be aligned with holes link

Figure 33: Servo motor aligned and mounted to link

5. Make set of wheels (see Fig. 34) and glue them to the bottom side of 3 links. Each of those 3 links should have one set of wheels, as shown in Fig. 34. The lego beams should be centered and as close to the edge of the link. This positions the wheels approx. in the middle of the link.

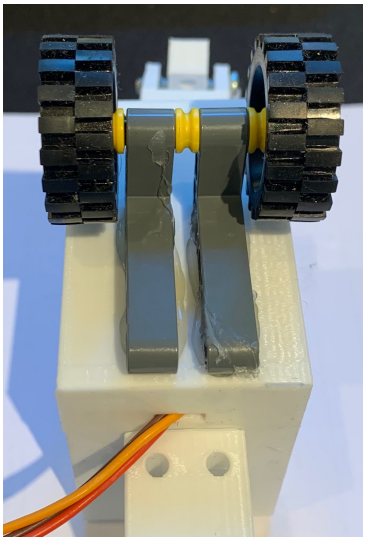


Figure 34: The lego beams should be centered, and positioned to the edge of the link.

6. Take a wheel, an axle 3, two long pins with friction and bushing and two technic beams. Shove the axle into the traces of the wheel, shove the two long pins onto the axle and attach the beams to either side. Glue the newly made set of wheels to a link. See Fig. 35 for reference.

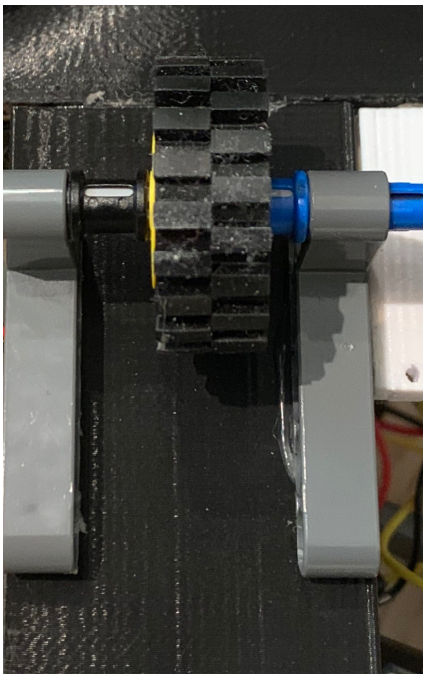


Figure 35

7. Take the remaining link and cut the top part (one with holes) in half with a mini-hacksaw. Then glue the fine side of cut part alongside the last link. Glue the compartment close to the edge of the cut part. Glue the face onto the link, so that the edge of the face is aligned with the edge of cut part. See Fig. 36 for reference.

8. Take the DC motor and stick it into the compartment. Keep in mind that the axle of the DC motor should go into the opening of the long pin. Now the dc motor is connected to the wheel and we have the head of the snake ready.

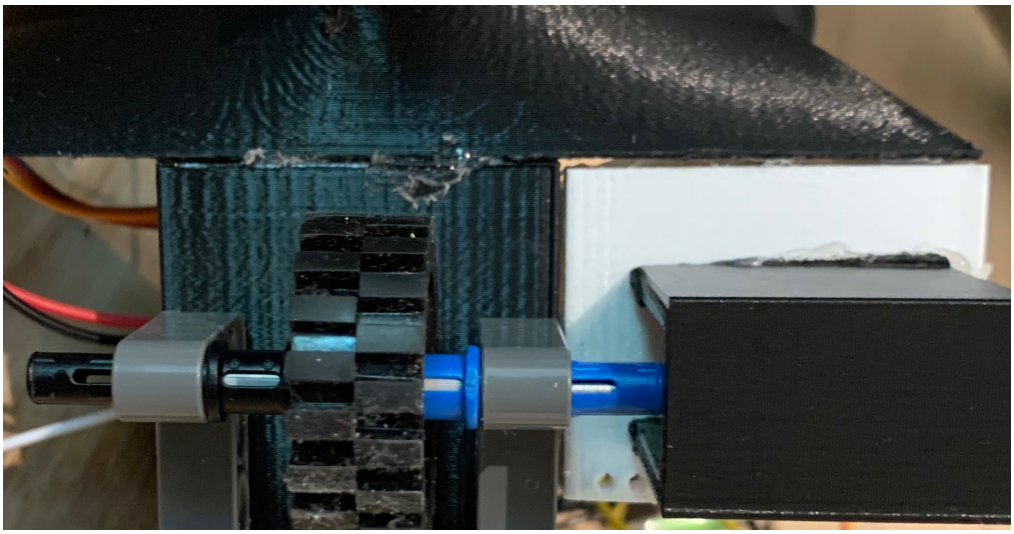


Figure 36

9. Take each link and screw the central bricks to each joint leg with M4 bolts. Now all links should have their set of wheels attached, servo motors mounted and connected to next link through joint. See Fig. 37 for reference. Note: Fig. 37 only shows 3 links, but you should have 4.

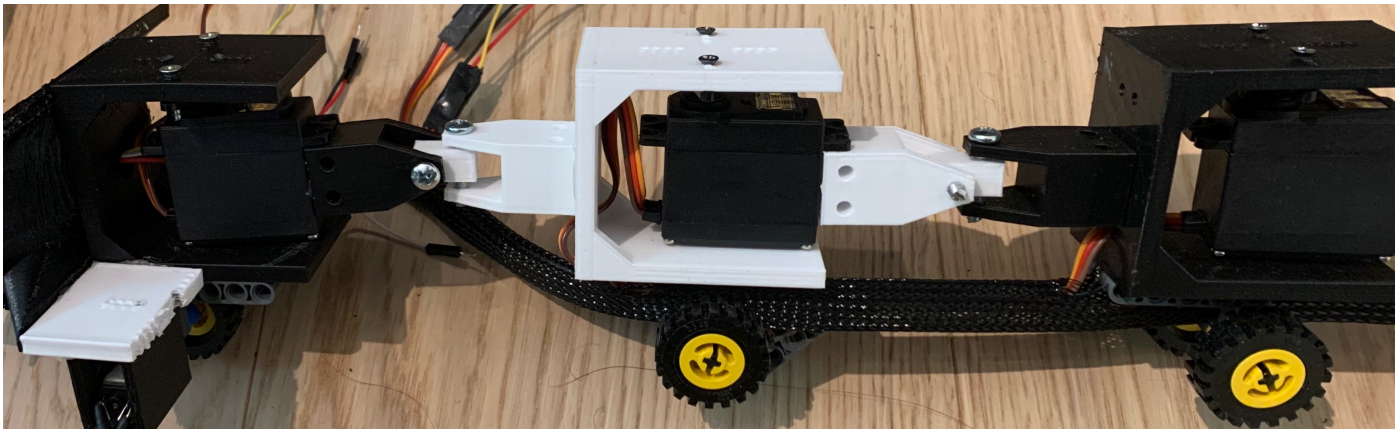
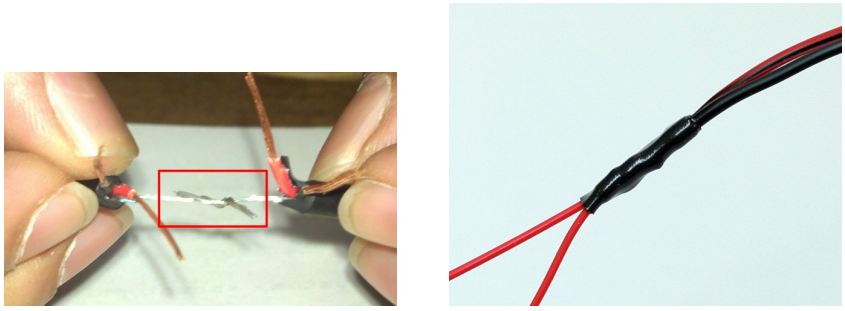


Figure 37: Head connected to 2 other links

10. Take electronic components, wire them as shown in Appx. C. In order to attach several components to the 5V output net of Arduino, clip the connecting cables, strip them until you have access to enough lead and twist the cables together. Tape the connection after in order to secure it enough. See Fig. 38 for reference.



38.1 38.2

Figure 38: Twisting wires and securing them with tape

11. Connect Arduino to your PC. For setting up your Arduino with your PC, see section 7.3. Upload the code in Appx. D to Arduino.
12. Done!

8.2 How to Operate Nagini

See section 3.2 in [1] for reference.

After bootup, Nagini will give user two possible modes of operation; one being autonomous mode and other manual mode. In autonomous operational mode, Nagini will try to follow an object to its best ability. In manual mode, the user can manually control Nagini with an infrared remote control. Here's the button configuration for manual operation:

- Move left - Button 4
- Move right - Button 6
- Move forward - Button 2
- Move backwards - Button 8

Figure 39: CarMp3 Infrared Remote

References

[1] H. Z. M. Singh S. Ahmed, "Nagini: Systems engineering management plan," University of South-Eastern Norway, Systems engineering management plan (SEMP), Jun. 2, 2020.

[2] ——, "Nagini: Project management plan," University of South-Eastern Norway, Project management plan (PMP), Jun. 2, 2020.

[3] S. Commanders, *Project scope statement*, 0.1, Feb. 14, 2020.

[4] ISO, *Systems and software engineering – software life cycle processes*, 2017, pp. 1–157.

[5] S.-n. Heo, H.-S. Lim, S. Hwang, and J. Lee, "Object following robot using vision camera, single curvature trajectory and kalman filters," in *Intelligent Robotics and Applications*, J. Lee, M. C. Lee, H. Liu, and J.-H. Ryu, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 562–575, ISBN: 978-3-642-40852-6.

[6] M. S. Sefat, D. K. Sarker, and M. Shahjahan, "Design and implementation of a vision based intelligent object follower robot," eng, in *2014 9th International Forum on Strategic Technology (IFOST)*, IEEE, 2014, pp. 425–428, ISBN: 9781479960620.

[7] E. Erzan Topçu, A. Demirkesen, and I. Yüksel, "Investigation of an object follower system," English, *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 24, no. 4, pp. 2412–2428, 2016, ISSN: 13000632.

[8] ISO, *Systems and software engineering - system life cycle processes*, 1st ed., ISO 15288, International Organization for Standardization, Case postale 56, CH-1211 Geneva 20, May 2015.

[9] M. [Babar], "Chapter 1 - making software architecture and agile approaches work together: Foundations and approaches," in *Agile Software Architecture*, M. [Babar], A. W. Brown, and I. Mistrik, Eds., Boston: Morgan Kaufmann, 2014, pp. 1–22, ISBN: 978-0-12-407772-0. DOI: https://doi.org/10.1016/B978-0-12-407772-0.00001-0. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780124077720000010.

[10] Wikipedia contributors, *Scrum (software development) — Wikipedia, the free encyclopedia*, [Online; accessed 29-May-2020], 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Scrum_(software_development)&oldid=959495566.

[11] ISO, *Iso/iec/ieee international standard – systems and software engineering - content of life-cycle information items (documentation)*, 1st ed., International Organization for Standardization, Case postale 56, CH-1211 Geneva 20, May 2019.

[12] S. L. Sohn. (Nov. 24, 2019). Progressive elaboration, [Online]. Available: https://www.projectmanagement.com/wikis/295452/Progressive-Elaboration.

[13] M. Singh, *Nagini: Master.ino*, Jun. 2020. [Online]. Available: https://github.com/Miniontab/Nagini/blob/master/Master.ino.

[14] ISO, *Systems and software engineering - life cycle processes - project management*, 1st ed., ISO 16326, International Organization for Standardization, Case postale 56, CH-1211 Geneva 20, Dec. 2009.

[15] P. M. Institute, *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*, 3rd, ser. PMBOK Guides. Project Management Institute, 2004, ISBN: 9781930699458.

[16] ISO, *Guidance on project management*, 1st ed., ISO 21500, International Organization for Standardization, Case postale 56, CH-1211 Geneva 20, Sep. 2012.

[17] ——, *Systems and software engineering — life cycle management — part 4: Systems engineering planning*, 1st ed., ISO 24748-4, International Organization for Standardization, Case postale 56, CH-1211 Geneva 20, May 2016.

[18] IEE, *Systems and software engineering—life cycle management—part 1: Guide for life cycle management*, 1st ed., IEE TR 24748-1, IEE Computer Society, 3 Park Avenue, NY 10016-5997, Jun. 2011.

[19] J. Andersson. (Jan. 20, 2020). Activities and life cycle models, [Online]. Available: https://www.dropbox.com/s/gc4akzbwtp9nu8y/T05.1%5C%20-%5C%20Activities%5C%20and%5C%20lifecycle%5C%20models.pdf?dl=0.

[20] ——, (Jan. 20, 2020). Extra network planning, [Online]. Available: https://www.dropbox.com/s/t7api1e42irtcpo/T05.2%5C%20-%5C%20Extra%5C%20Network%5C%20planning.pdf?dl=0.

[21] A. Shenhar and D. Dvir, *Reinventing Project Management : The Diamond Approach To Successful Growth And Innovation*. Harvard Business Review Press, 2007, ISBN: 9781591398004. [Online]. Available: http://ezproxy2.usn.no:2056/login.aspx?direct=true&db=nlebk&AN=675086&site=ehost-live.

[22] D. Workshop. (Feb. 10, 2018). Stepper motors with arduino - controlling bipolar & unipolar stepper motors, [Online]. Available: https://youtu.be/0qwrnUeSpYQ.

[23] ——, (Jul. 14, 2018). Getting started with lidar, [Online]. Available: https://youtu.be/VhbFbxyOI1k.

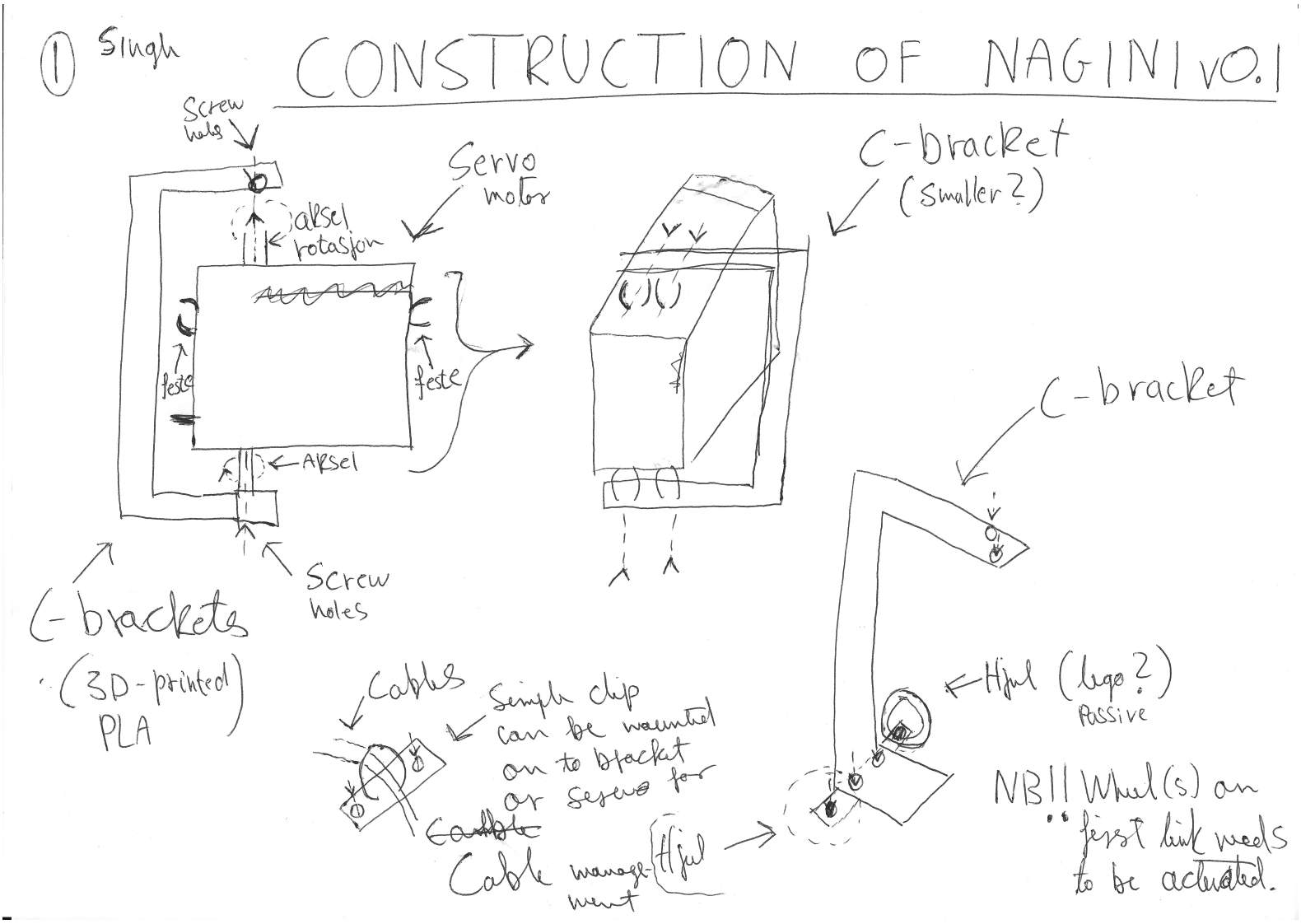
[24] P. McWhorter. (May 31, 2019). Arduino tutorial 1: Setting up and programming the arduino for absolute beginners, [Online]. Available: https://youtu.be/fJWR7dBuc18.

[25] Scrum.org. (May 30, 2020). What is scrum? [Online]. Available: https://www.scrum.org/resources/what-is-scrum.

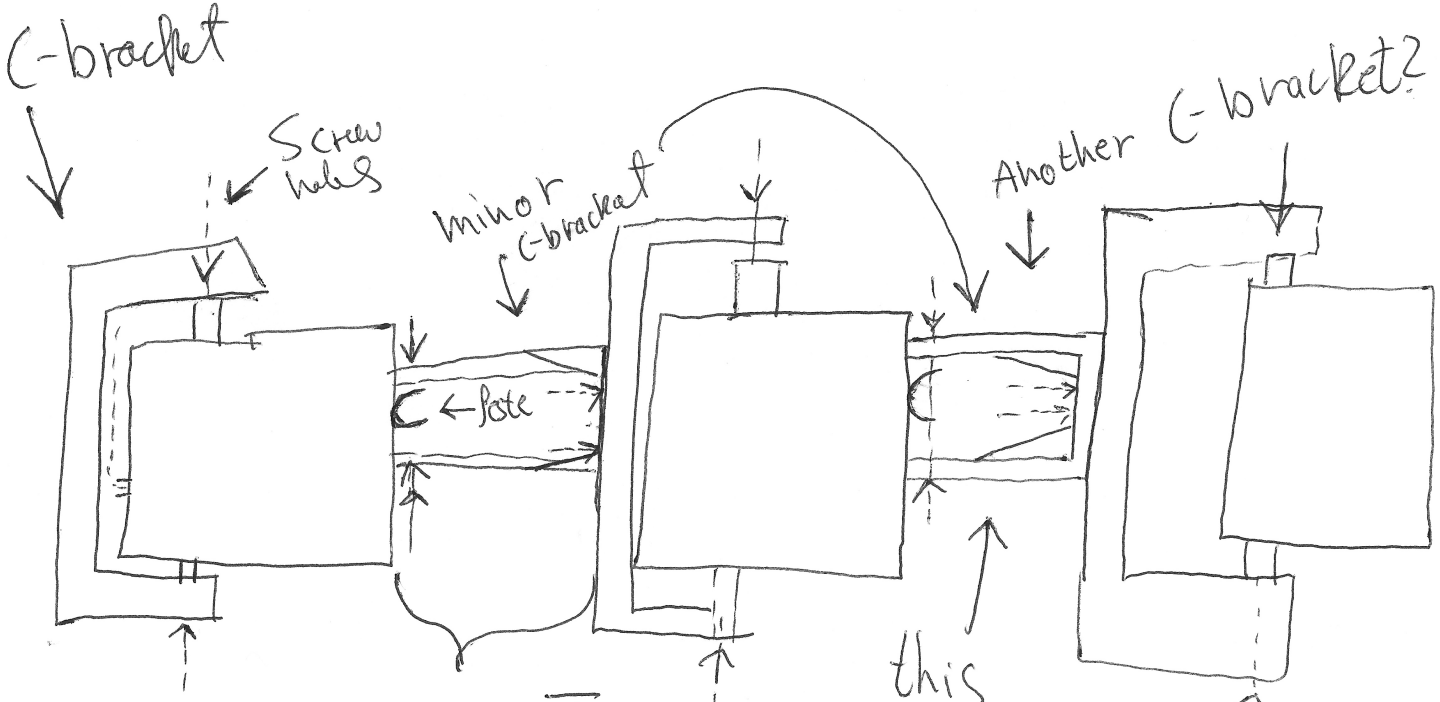
[26] K. Schwaber and J. Sutherland. (2017). The scrum guide™, [Online]. Available: https://www.scrumguides.org/scrum-guide.html.

Appendices

Appendix A Construction plan of Nagini v0.1

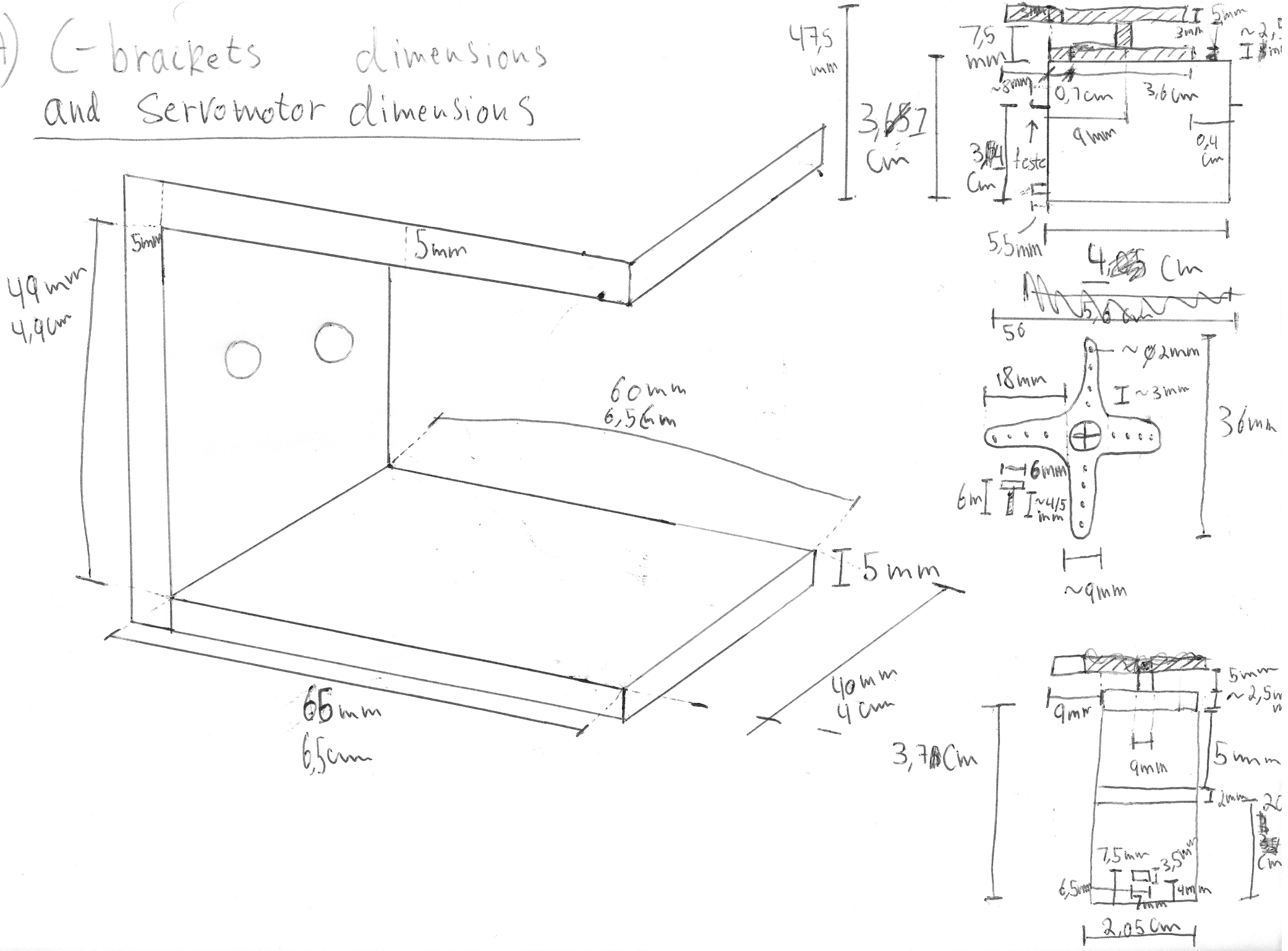


(2) Singh ~~TOTAL~~ LENGTH



2A) C-brackets dimensions and Servomotor dimensions

Singh

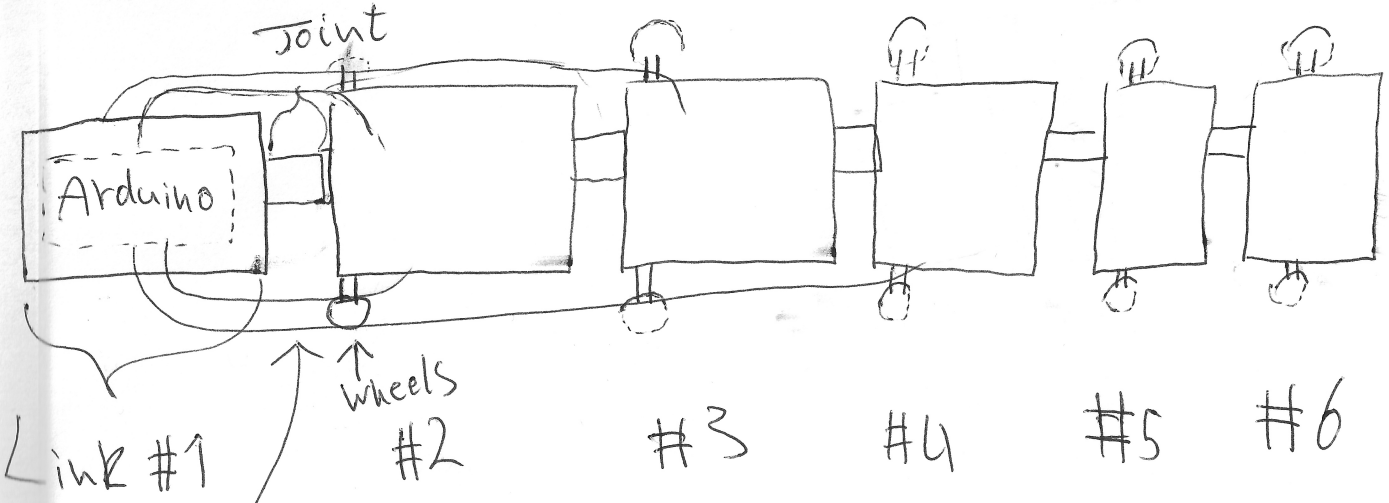


(3) Singh

Top-View of Nagini

Arduino can be ~~W~~ velcro'ed on top or in link ~~#~~#1 ~~#~~ with a easy 3D-printed case on-top for protection and access. Case can have holes in it for wire management

~~S~~Nagini head

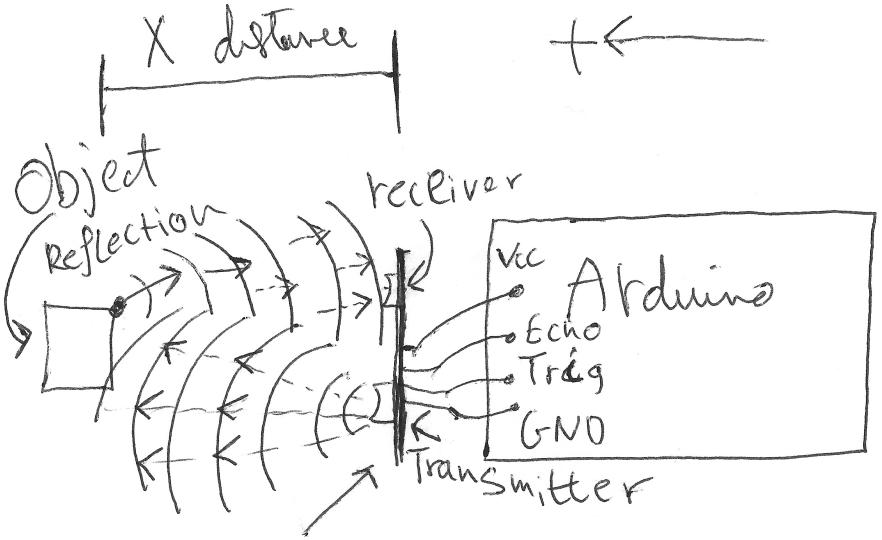


Wires can go on top of the model through clips (page 7).

4 Sarah

Detection Subsystem

A

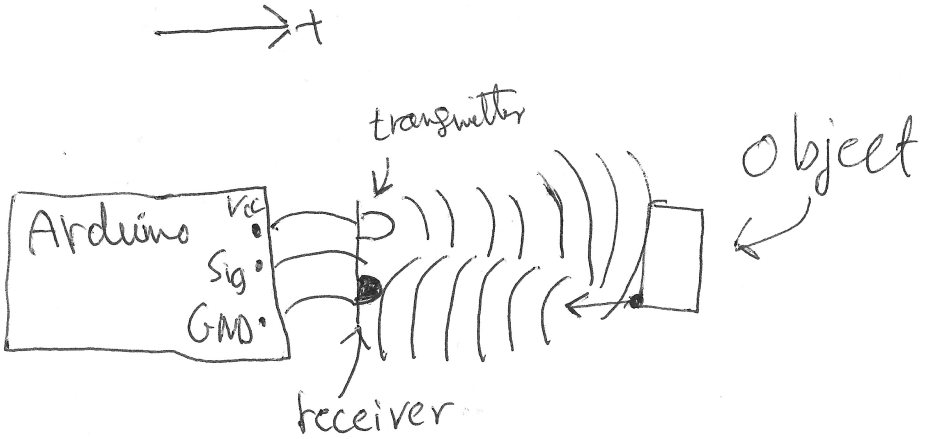


Ultrasonic Sensor HC-SR04

Time between transmission and reception of signal can help us calculate distance ~~between~~ to object, because we know V_{sound} in air.

To follow the object, the robot has to travel X-distance

B

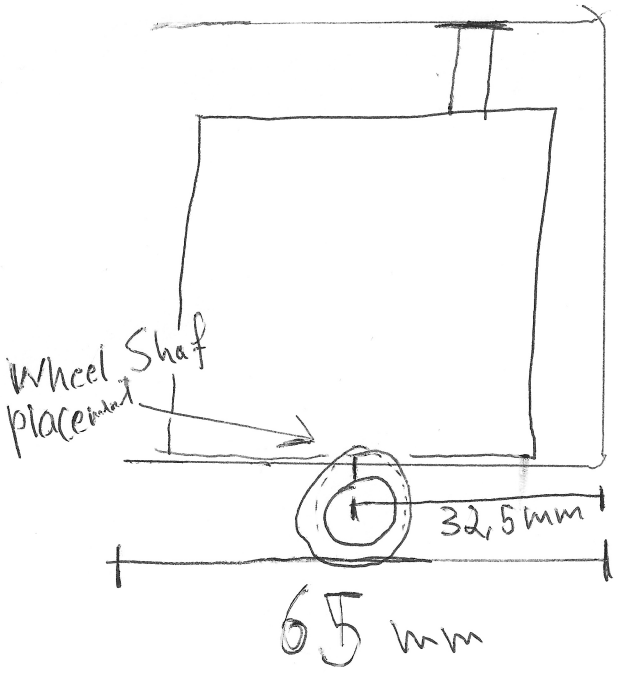


IR Obstacle avoidance Sensor

The IR transmitter sends IR light, and the receiver gets the reflected signal back and understands that there's an object there.

(5) 3.3.2020
Seigo and Samuel

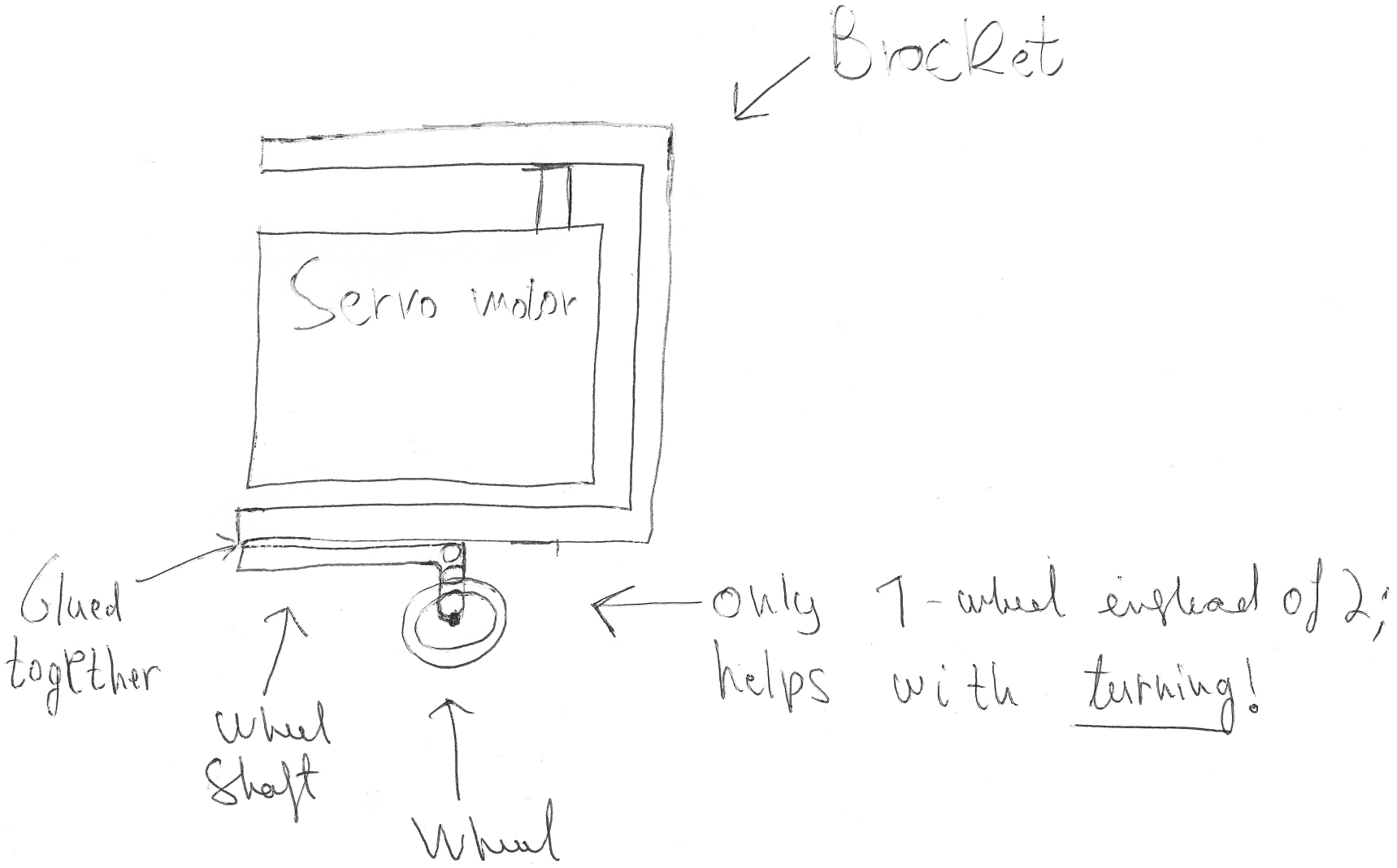
Wheel- and shaft placement



Problem! When servo rotates 90° left or right, it hits the wheels!
See (6) for alternate wheel solution!

(6) 03.03.2020
Singh and Samuel

Re-did Wheels because 1. pair of wheels were obstructive.



Singh

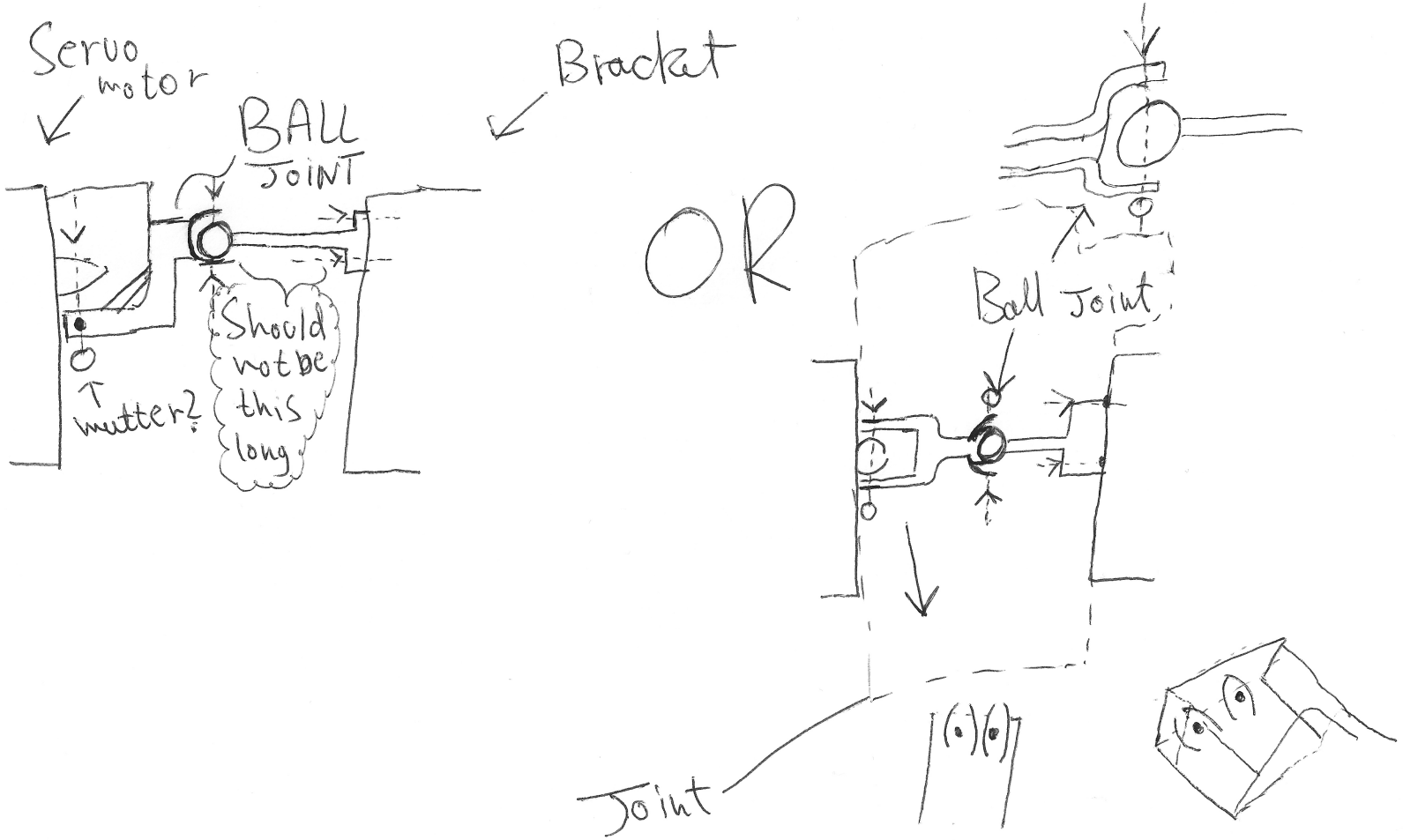
Things to do - 03.03.2020 17:12

- Hurry manufacturing of prototyping parts since Richard isn't available rest of week 10.
  - Laser cut rest of prototyping parts ✓
    - Joint
    - Link
  - Build prototype Nagini with laser cut parts.

- ~~Design Joint - 3D~~
- Design snake head with room for Arduino ~~uno~~ (batteries?)
- 3D-print ALL parts when Richard is back

(7) Singh

New type of Joint (Ball Joint)

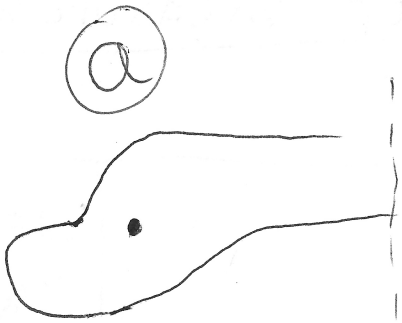


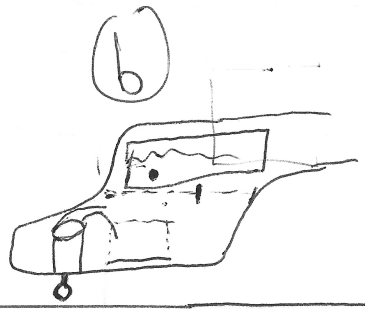
This new joint ~~is~~ grants freedom of movement to link $_{n+1}$, unlike Joint V.1.

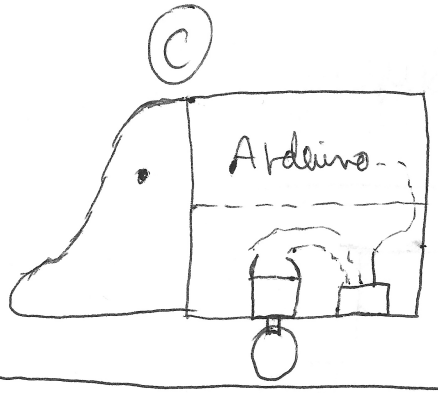
09.03.2020

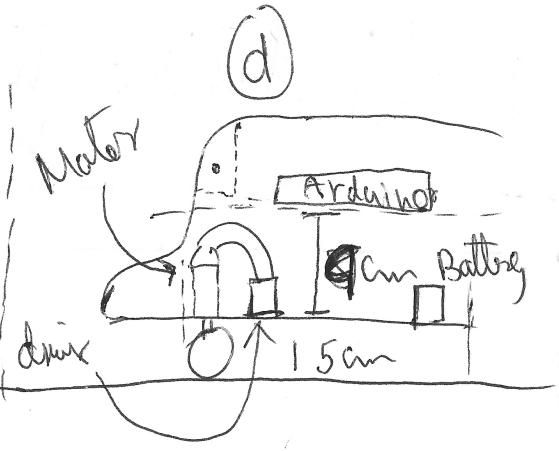
Snake Head Model

Samuel and Singh

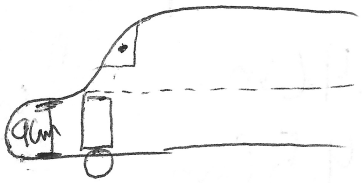


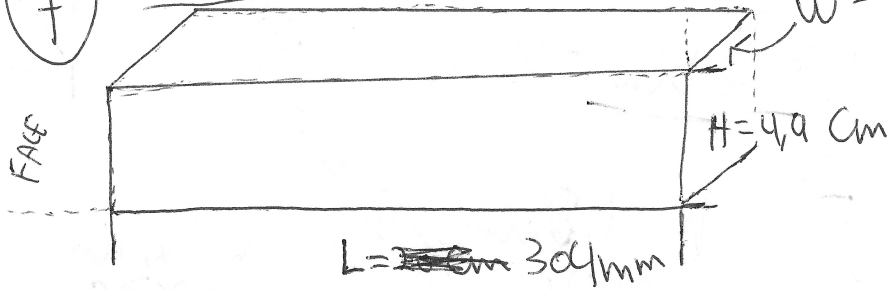


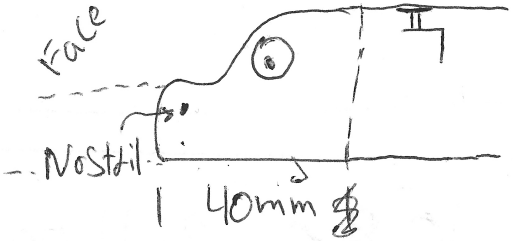




Side View





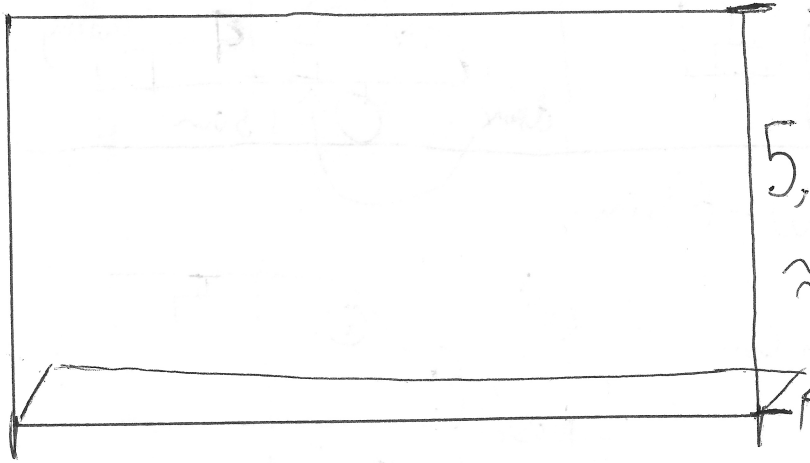


L= ~~1,9 F 20 cm~~ , W = 6,5 cm , H = Same as bracket, ☐ 4,9cm

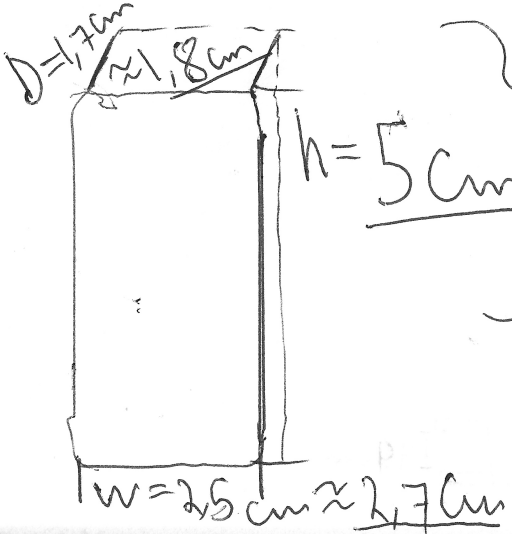
8a Singh and Samuel

Measurements of Various components

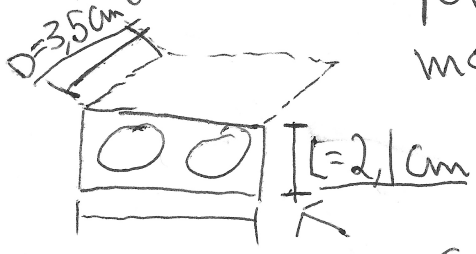
Arduino measurement



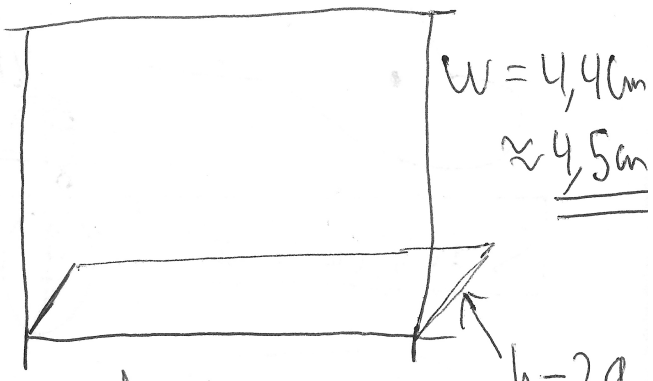
l= 10,8cm ≈ 11,00cm

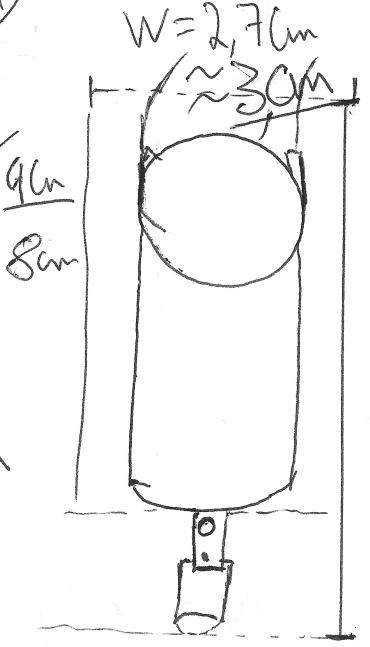


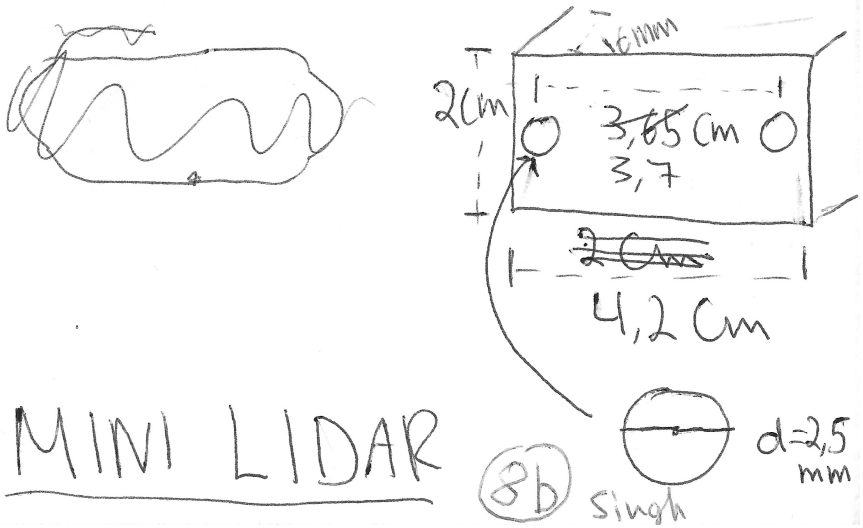
Battery



Motor driver measurements







MINI LIDAR

8b

Singh

10.03.2020 | Singh

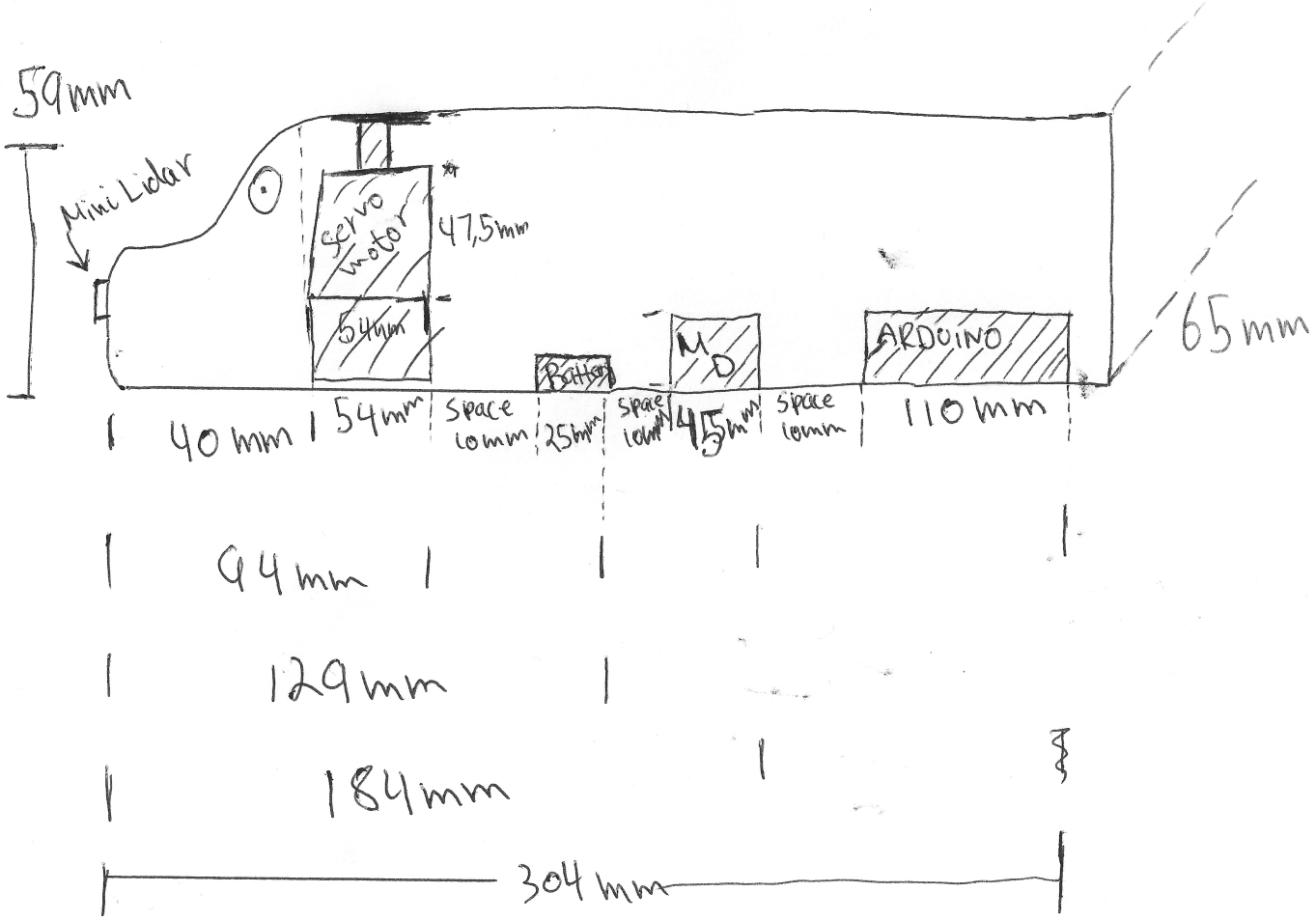
1. Snake head → 3D print!
2. Lower part of snake head with movement subsystem
3. Connect everything

10.03.2020

(9)

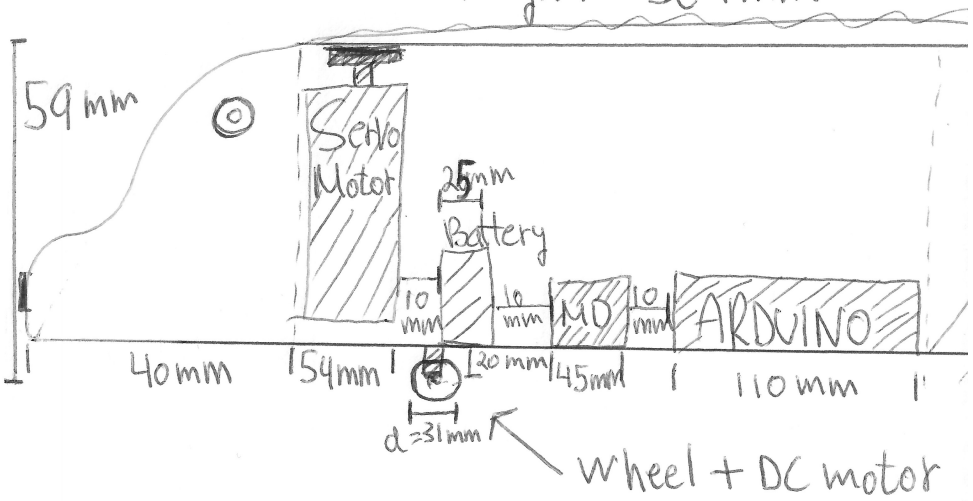
Snake Head W/ Components

Singh and Samreed



9a Singh

Snake height = 59mm
width = 65 mm
length = 304 mm



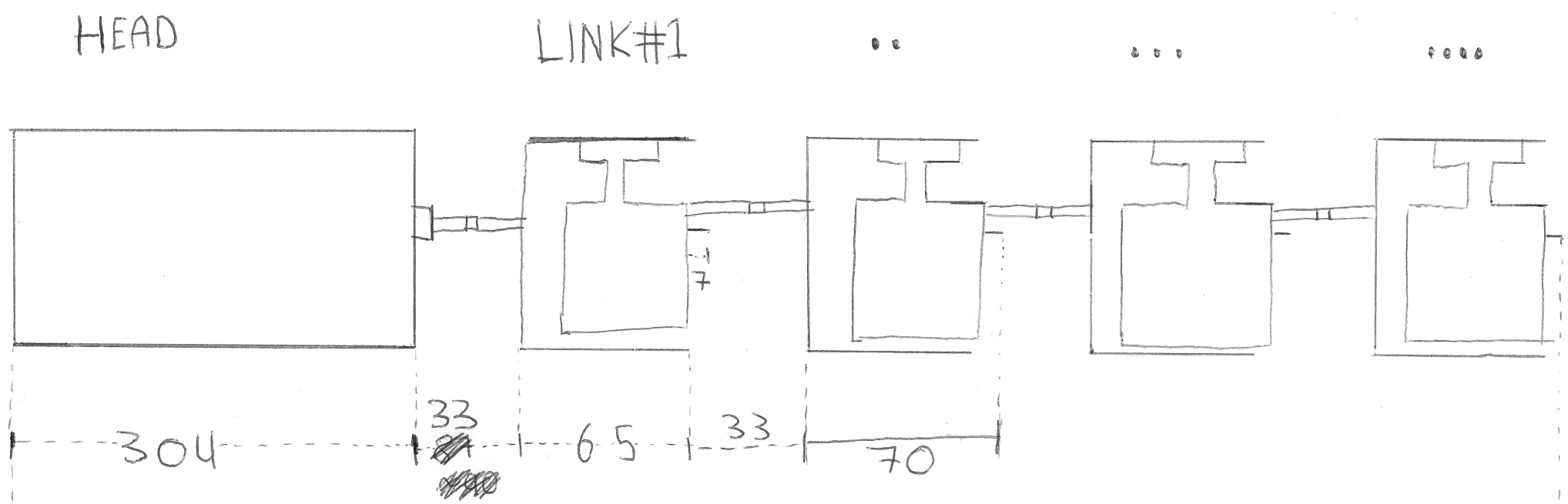
304 mm

~~304~~ mm

Fri 13.03.2020 (10)

Overview of Nagini 0.1V

Singh

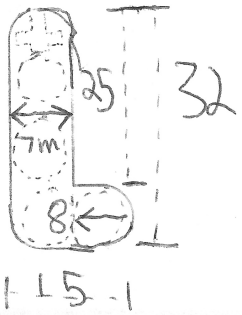


~~924 mm~~ 696mm

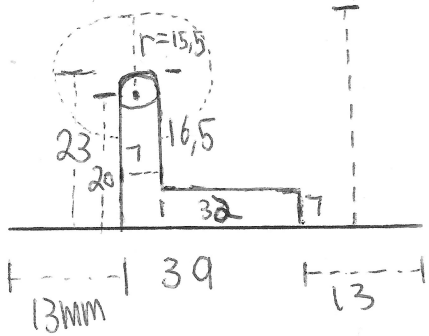
~~92,4 cm~~ 69,6 cm

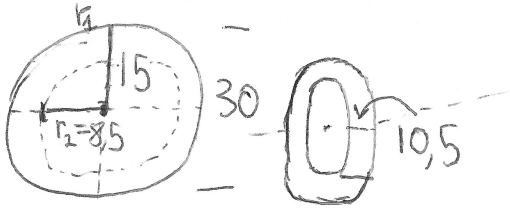
~~~~ ~ 70 cm

0,7m


~~~~

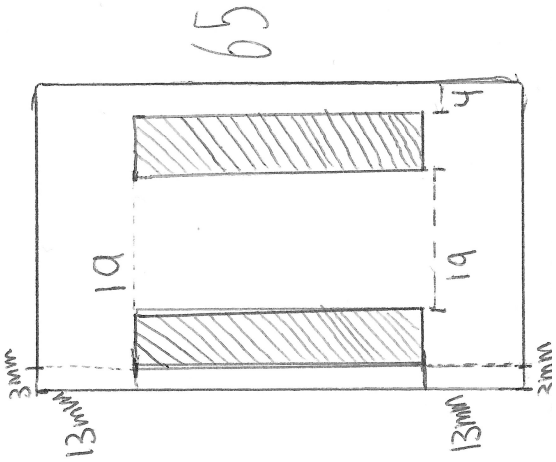
ALL UNITS
IN mm = 10·cm

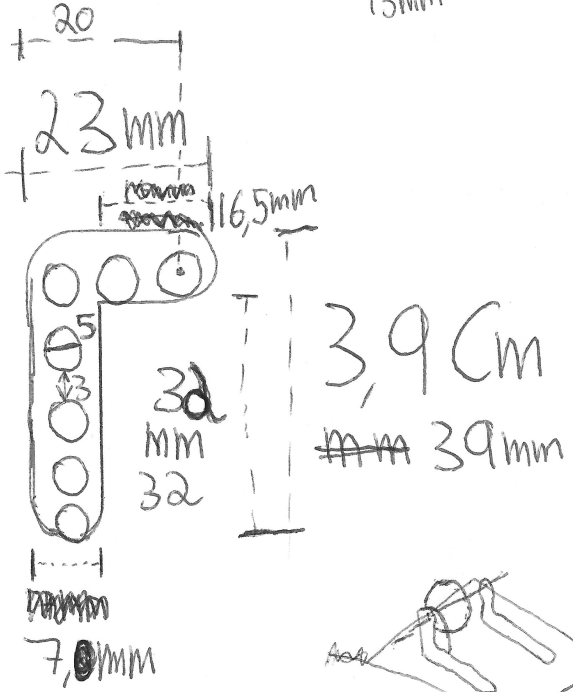


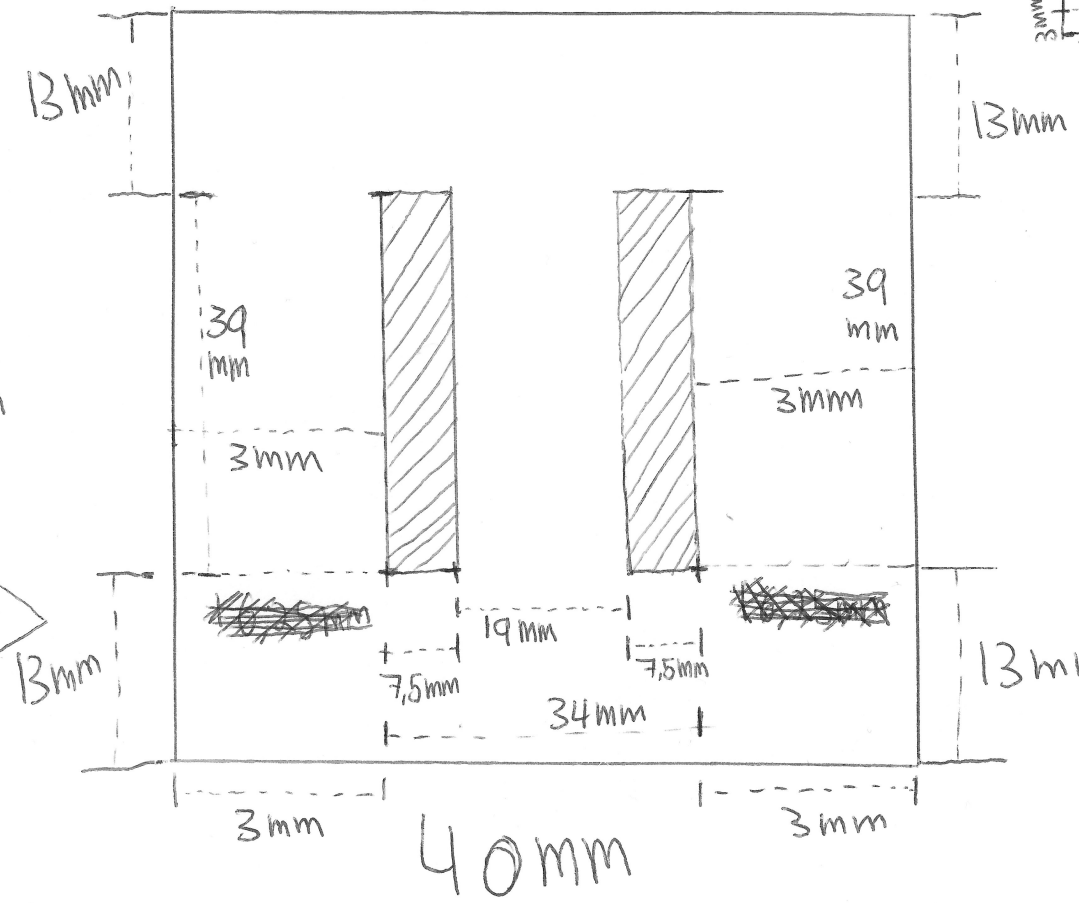


$r_1 = 15$
$d_1 = 2r_1 = 30$

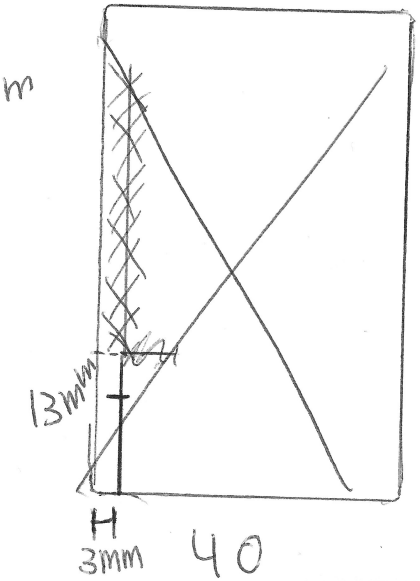
$r_2 = 8,5$
$d_2 = 2r_2 = 17$







65 mm

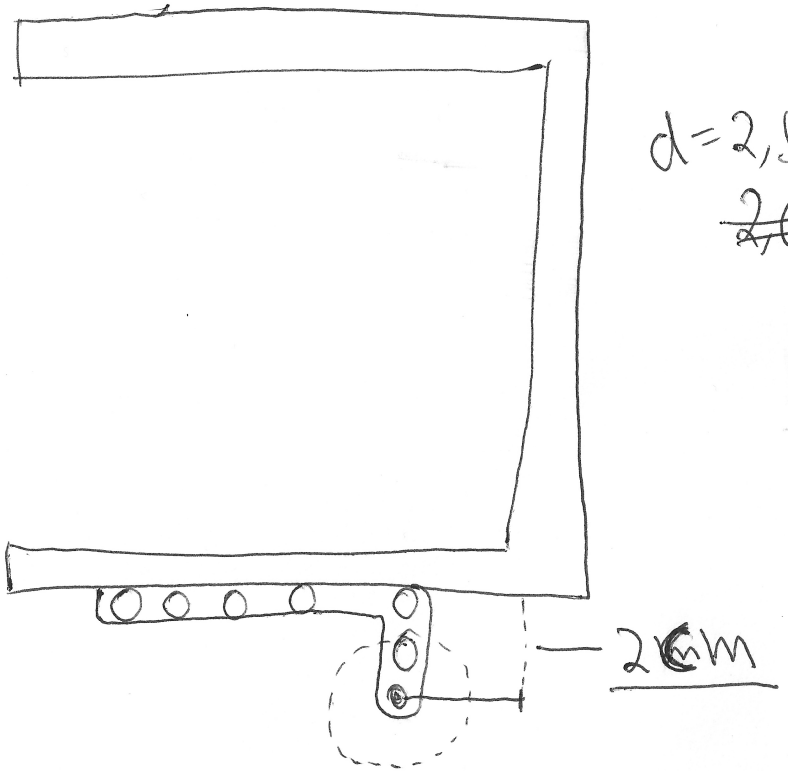


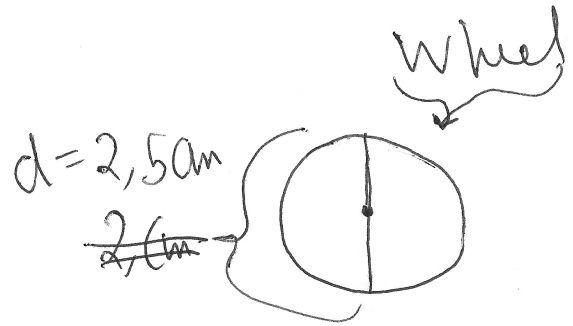
Wheels and axle placement

SINGH
14.03.20 02:45 (11)

Bottom view of C-bracket

11a Singh

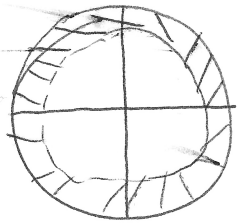




11b

Wheels

Singh and Samuel



d = 3,1 Cm

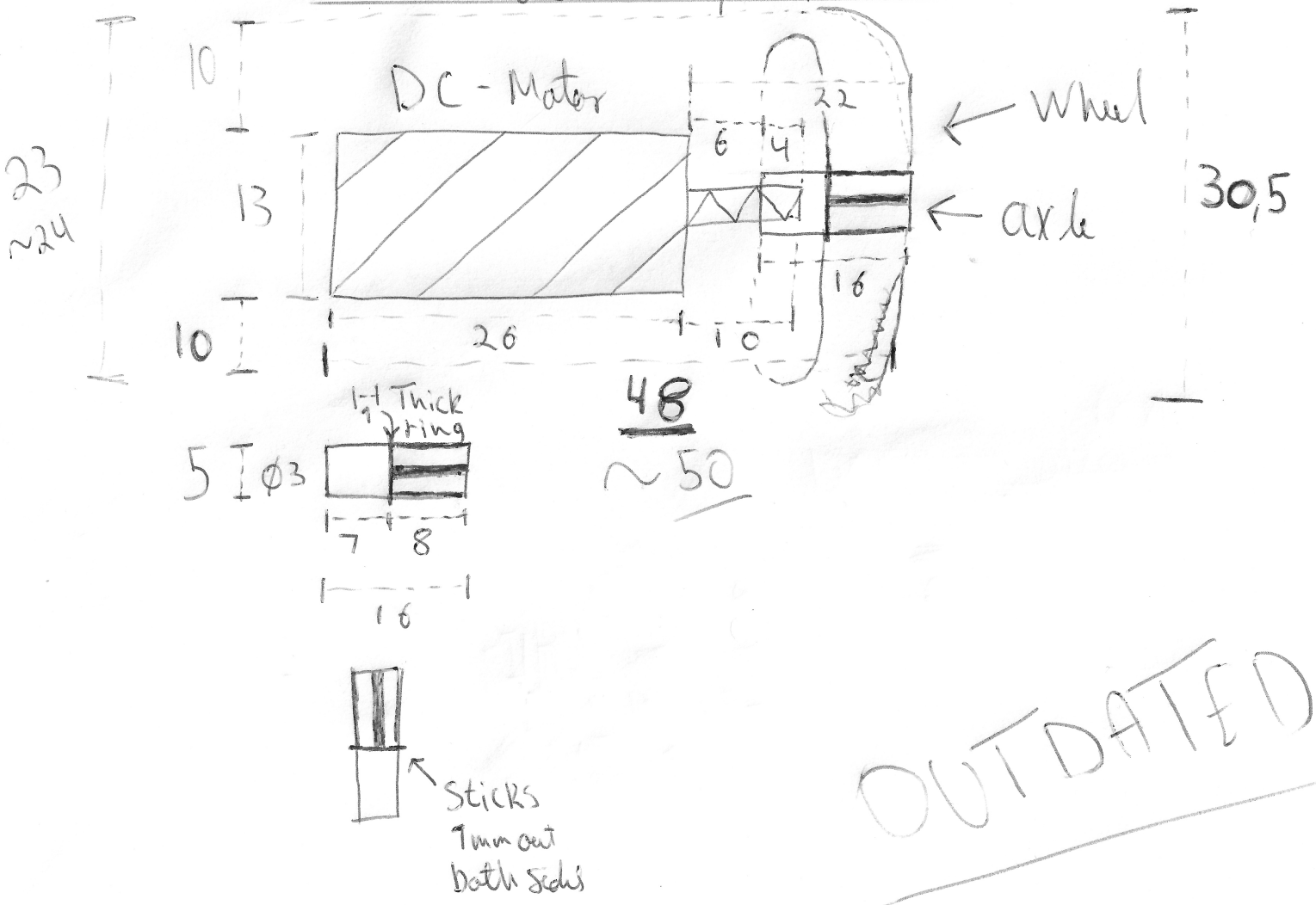
r = d/2 = 1,55 Cm

16.03.2020 Singh

(12)

Singh

DC-Motor w/ axle and wheel



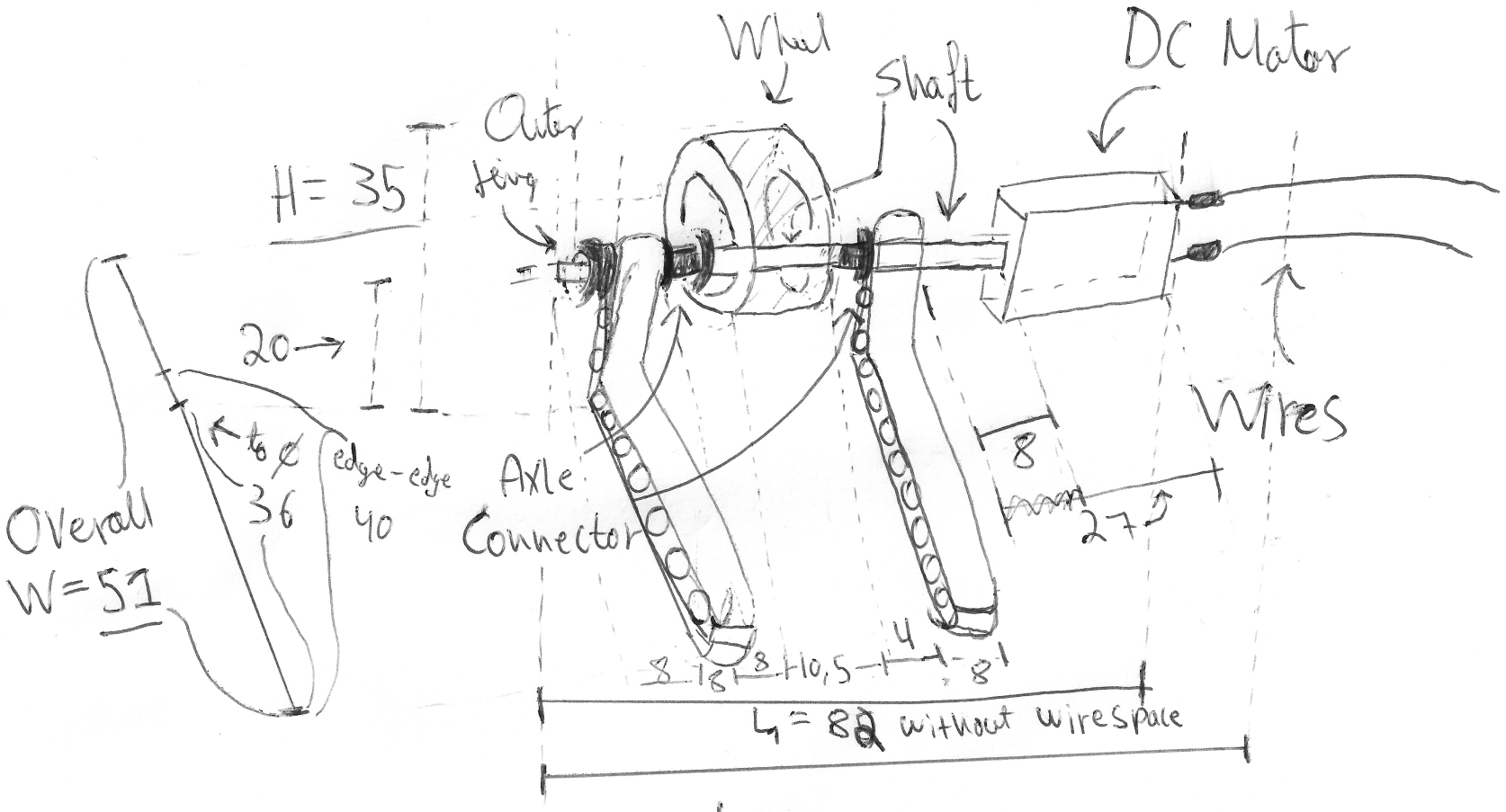
OUTDATED

17.03.2020
Sleigh

ALL UNITS [mm]

(13)

Movement Subsystem - DC motor w/ wheel

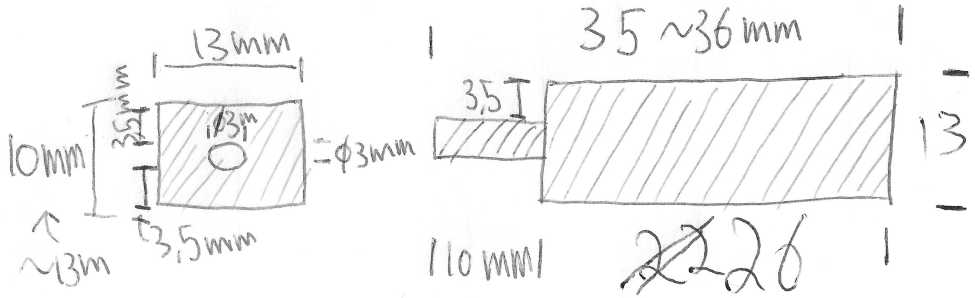


$L_2 = 105$ mm
w/ wire space

Singh

13a

DC-MOTOR

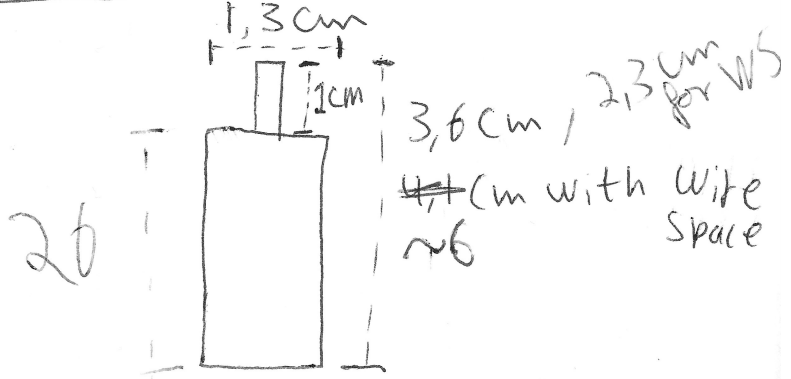


The ~~motor~~ is near ~~quadr~~ quadratic!

Singh

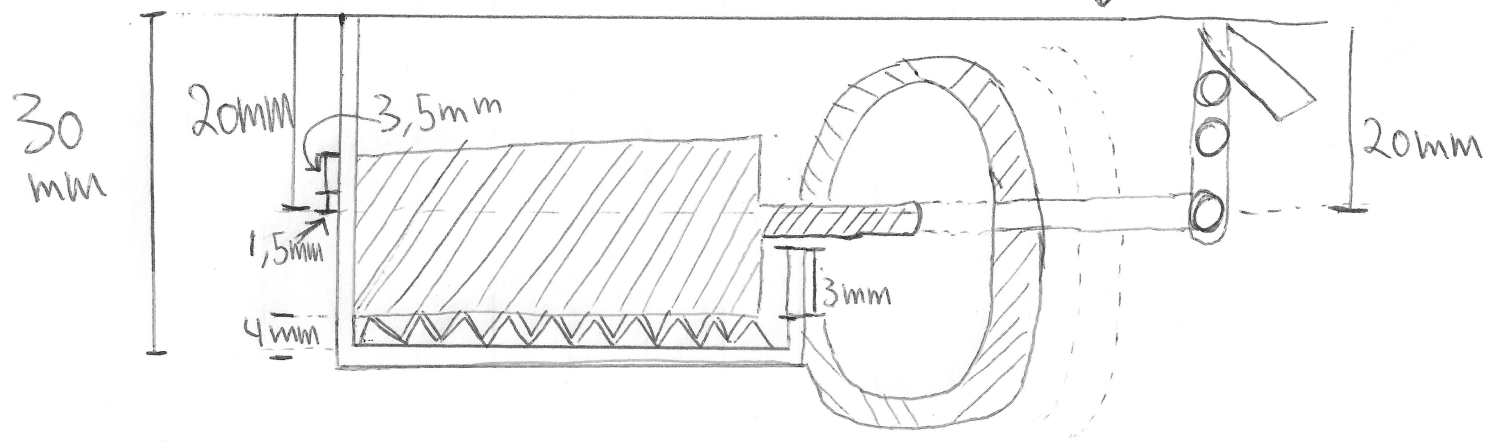
13b

~~DC Servo~~ DC Motor



13C
Singh

Maybe 2 wheels?



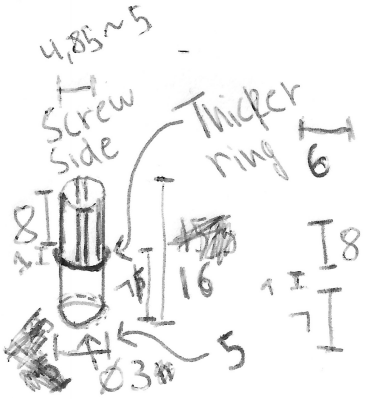
(10mm)

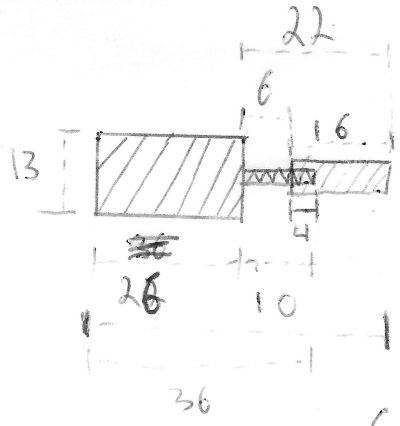
Velcro

The water compartment will need support to baseline on both sides?

13C.1
Singh

ALL UNITS IN mm.

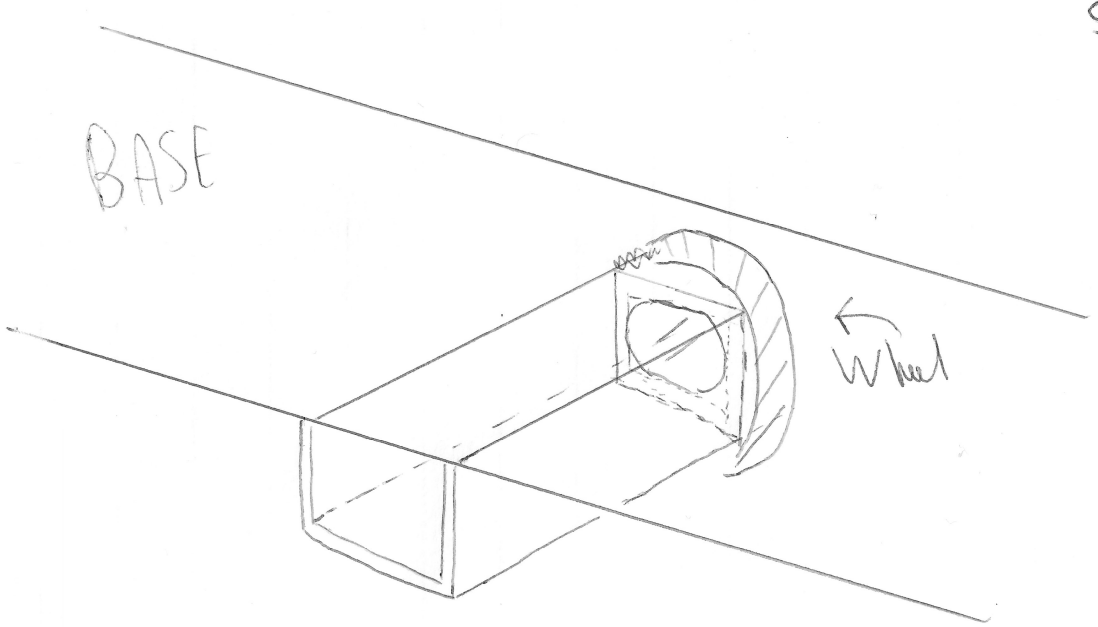


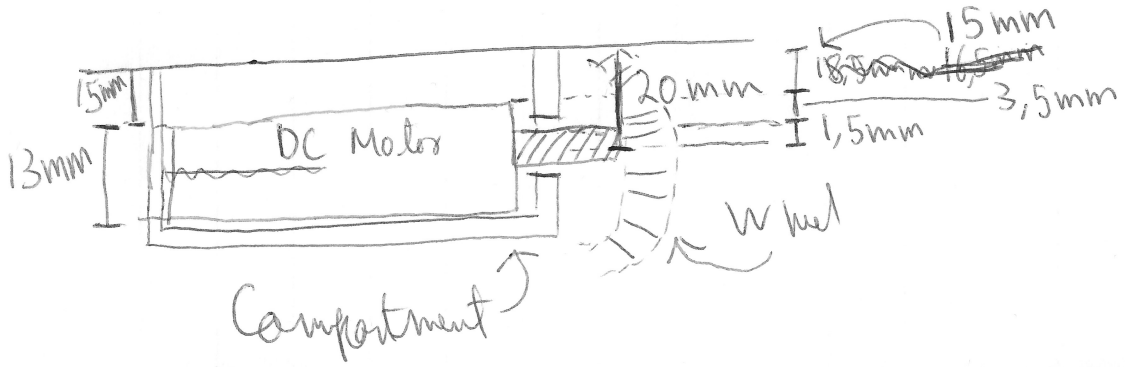


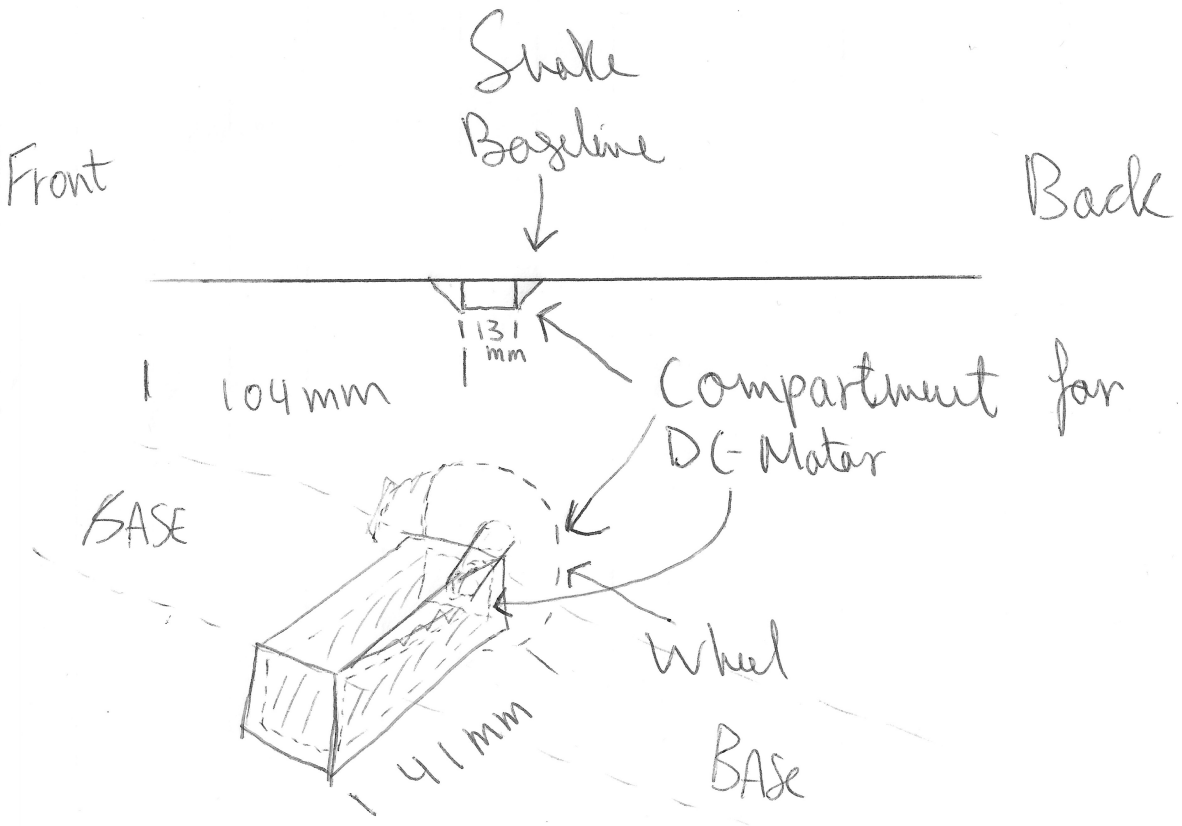
(axle)

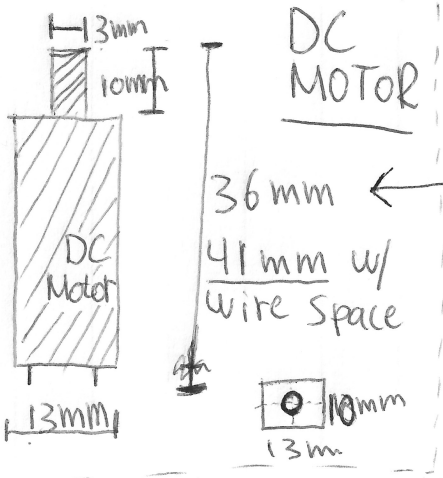
DC-Motor w/shaft

Singh (13d)


BASE
Wheel


15mm
13mm
DC Motor
20mm
15mm
3,5mm
1,5mm
Wheel
Compartment


Snake
Baseline
Front
Back
131 mm
104mm
Compartment for
DC Motor
BASE
Wheel
41mm
BASE



DC MOTOR

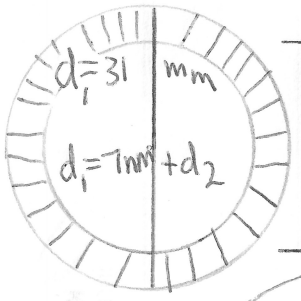
36mm

41mm w/ wire space

10mm

13mm

Wheel



d_2=30mm

r=15mm

w=10,5 mm

W=65mm ← CHANGE TO 75mm → PROBLEM SOLVED!

L(DC) > W/2

BACK

32,5 mm

Wheel

27,25mm | 10,5mm | 27,25mm

DC Motor

l=304mm

Ø=4mm Holes for wires through

36 mm

104mm

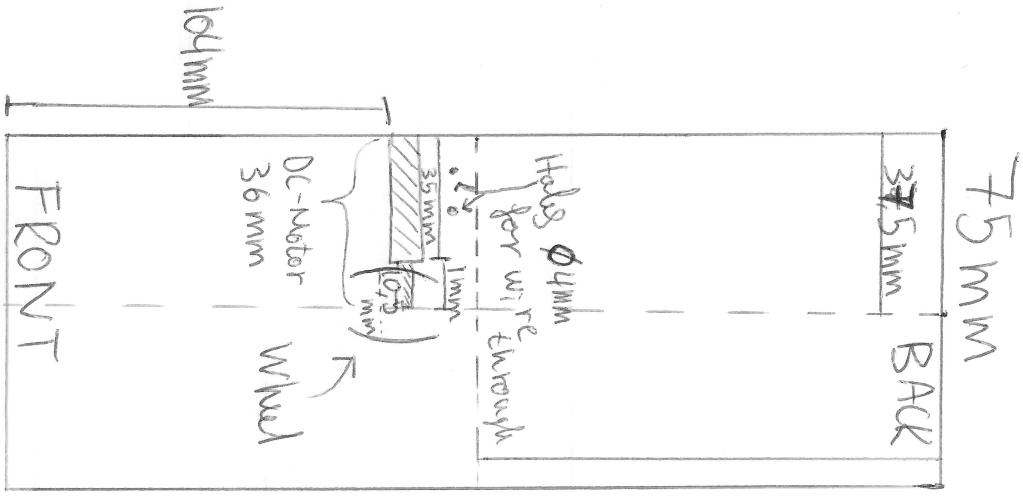
FRONT

DOES!!! IT NOT ... WORK, DC MOTOR LENGTH > $\frac{W}{2}$

Shake bottom view

13e.1 Singh

If we change Snake width to 75, problem Solved!



BOTTOM VIEW

304 mm

13e.2

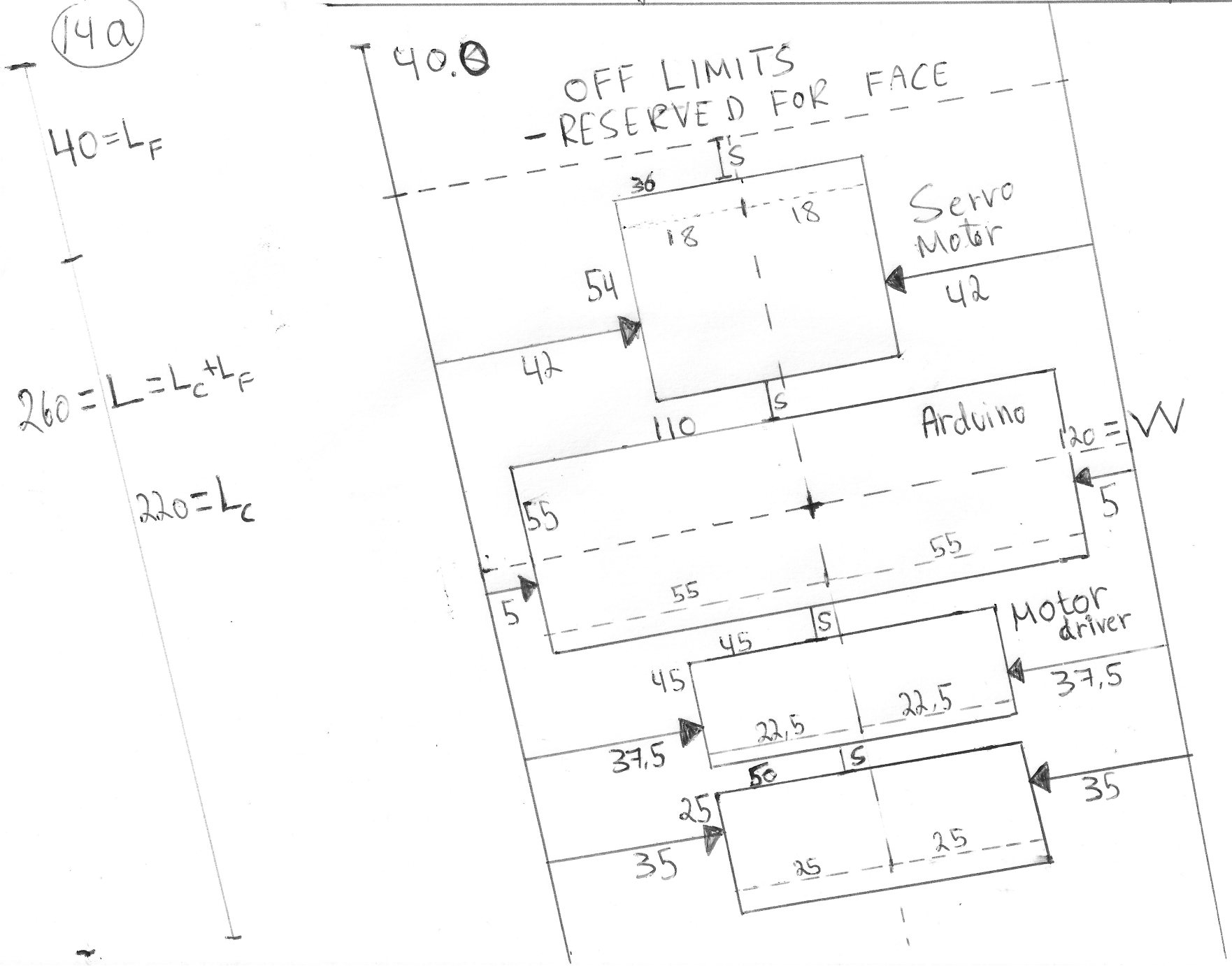
Singh

18.03.2020
Singh

ALL UNITS in mm

Placement of Components in Nagini's Head

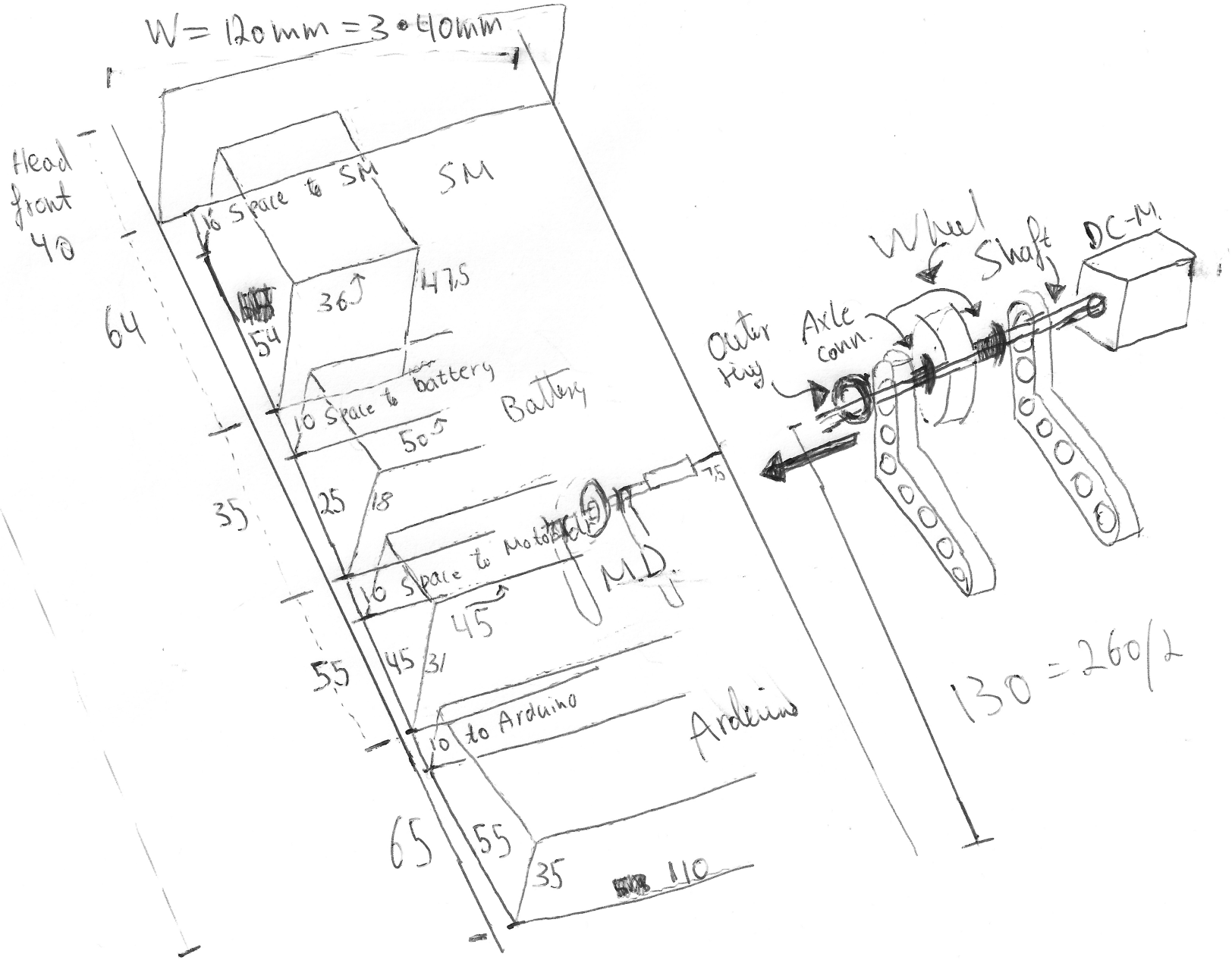
(14a)



18.03.2020
Singh
14b

ALL UNITS [mm]

Nagini's head w/ movement subsystem

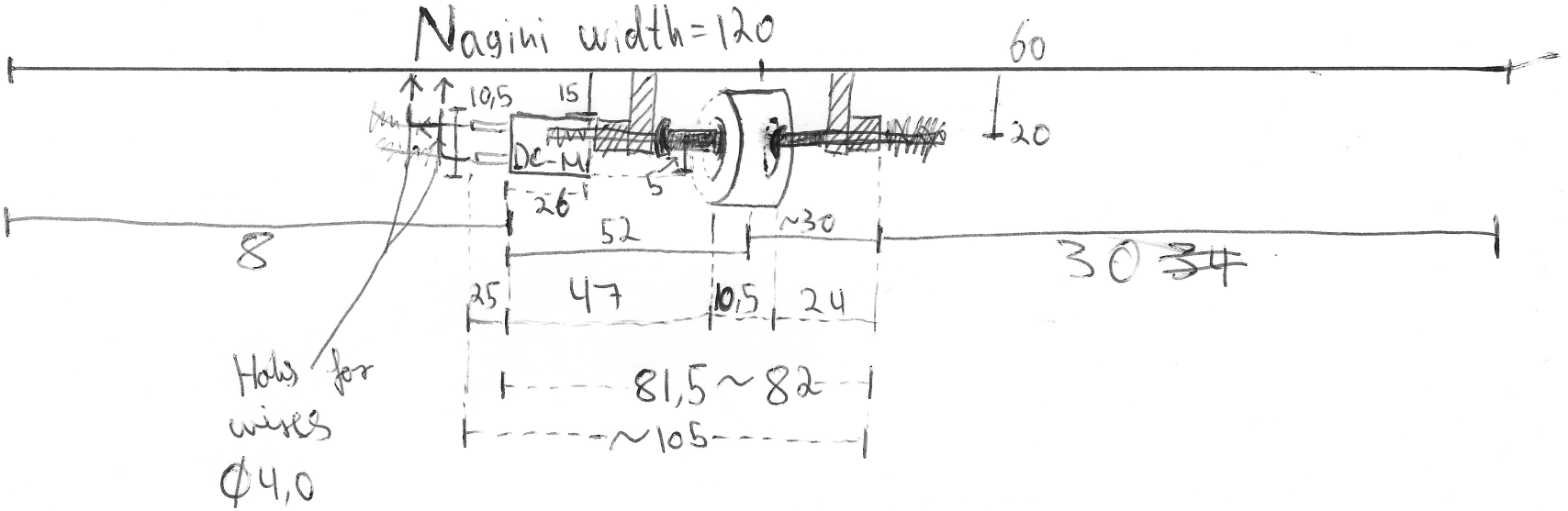


~~344~~
260

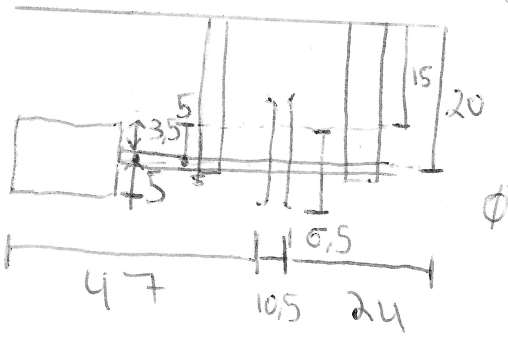
19.08.2020
Singh

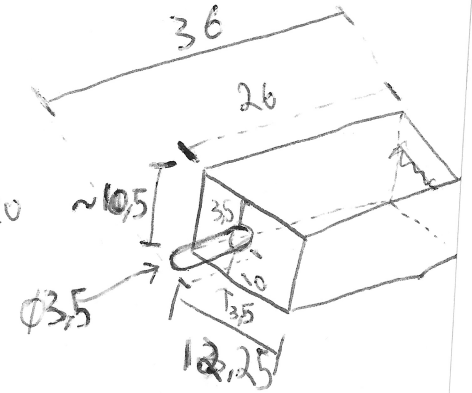
Movement Subsystem placement

(15)



(15a)





DRAFT

Singh and Samuel

Total Components of Nagini V0.1

- 1x Arduino Mega
- 1x 9 or 12V Battery
- ~~12~~ x DC motors (12VDC, 300rpm)
- 1x Motor driver
- 1x Mini Lidar
- ~~1x Wheels for Snake head~~
- C-brackets (Link) X 4
- Joint x ~~8~~ 4
- Snake head (1 link) X 1

} 5 LINKS

- 5x Servo motors
- 5x Wheels

Appendix B Total equipment

Electronics:

- Arduino Mega 2650
- L298N Motor Driver
- N20 Micro DC-Motor (12V - 300rpm)
- 9V Lipo battery
- VS1838B IR Receiver
- IR Remote
- Seed Mini Lidar/HC-SR04 Sensor
- Hitachi HD44780 compatible LCD w/ I2C backpack
- 4x - TowerPro MG995 Servo Motor
- Half-sized breadboard
- Dupont jumper wires

3D printed parts:

- 3x - Links v0.2
- 3x - Joints v0.4
- 3x - Central Brick v0.3
- Head v0.2
- Face v0.1

Lego modules:

- 6x - Technic Axle 6
- 14x - Technic Beam (90 deg)
- 2x - Long Pin
- 10x - Wheel Hub
- 10x - Narrow Tire with Ridges Inside
- 21x - Half Bushing

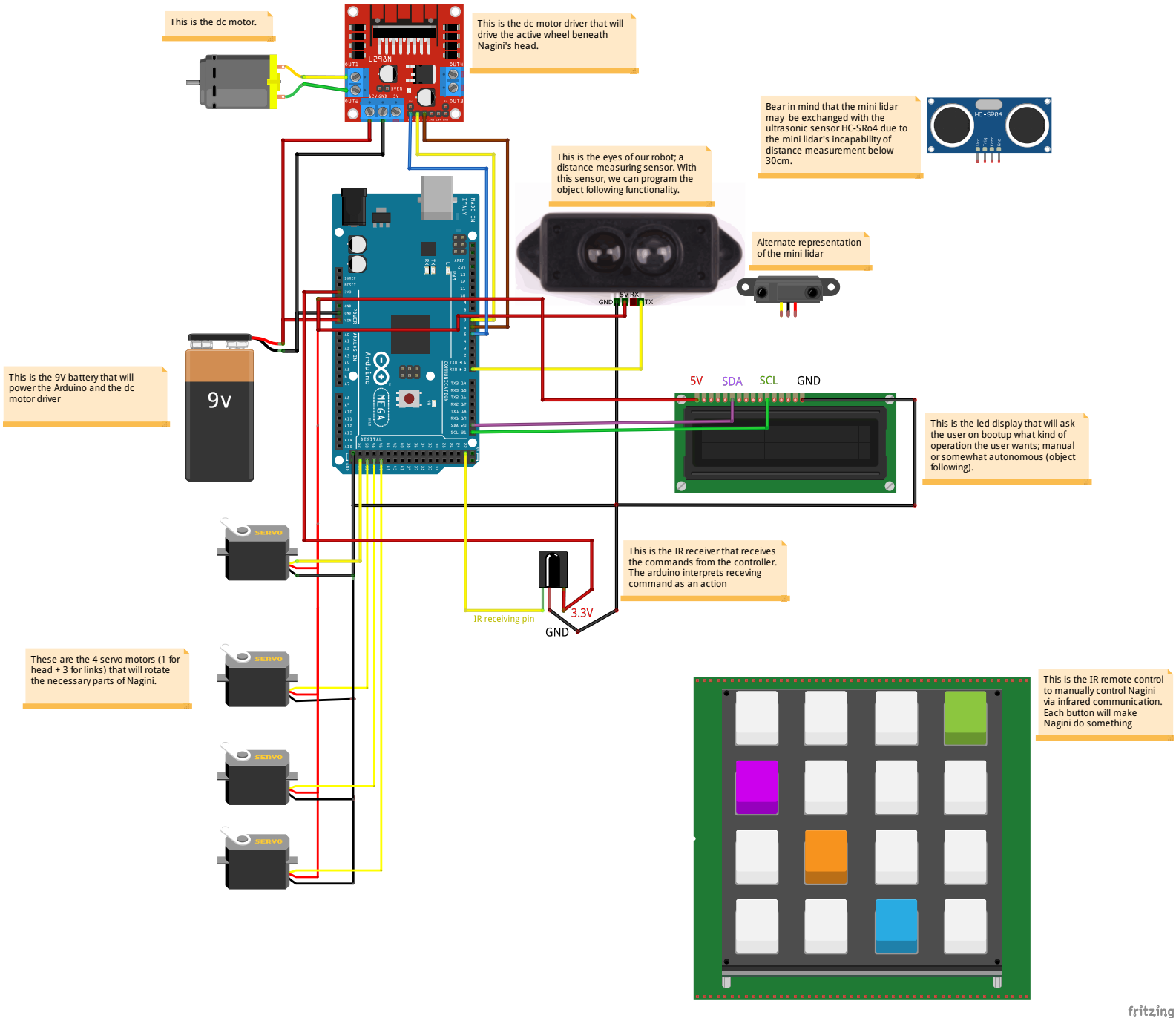
Screws/bolts:

- 12x - M3 (3x10mm) flathead woodscrews
- 8x - M4 (3.8 to 4x25mm) bolt/screw

General equipment:

- Cable sleeve (Ø: 12mm)
- Double-sided tape
- Velcro
- Glue gun
- Screwdriver set
- Drill with accompanying drill bit set

Appendix C Arduino Connectivity Diagram



Appendix D Master code for Arduino

To access this code in Github, see [13].

```
/* Author: Mehtab Singh
                                                                    PLEASE READ
    BEFORE PROCEEDING FOR CODE

    This master source file incorporates software to control Nagini v0.1.
    An accompanying ASMD chart explaining the behaviour of this software can be found
     at https://www.dropbox.com/s/i791sz6ga2d7ejx/ASMD.jpg?dl=0

    Few words about system-of-interest (more info found in referenced documents above)
     :
    The system-of-interest is a robotic snake that consists of 4 servo motors (and a
     motor driver) for turning s-o-i,
    a DC motor (12VDC-300RPM) for forward propulsion (actuating a single wheel beneath
      snake's head),
    a LCD screen for displaying messages,
    an IR (infrared) sensor and accompanying remote for IR communication
    and a HC-SR04 sensor for measuring distance to an immediate object.
    This allows for a system that can follow an object (Autonomous mode) or to control
      the snake manually (type of motion is lateral undulation or serpentine movement
     )


    The structure of this software:
    1. Including libraries, defining/declaring or instantiating necessary variables or
      component objects,
    2. In setup: Initializing components and code as necessary
    3. In loop: Magic happens.

    The behaviour of this software:
    1. At initialization, the LCD screen will ask user to choose a operational mode
     for system-of-interest; that is either Autonomous or
       Manual (which is manual control of Nagini with serpentine motion). The user
     sets this with key press on the infrared remote; pressing the button with '1' on
     it will make the
       system-of-interest autonomous, while any other key press is manual control.

       A figure depicting the infrared(IR) remote control codes for used IR remote (
     CarMP3: http://www.geeetech.com/wiki/index.php/Arduino_IR_Remote_Control).
       Credits are due where credits are deserved (for the following figure): https://
     github.com/steakknife
          ____________________
        /                      \
       |  CH-     CH     CH+   |
       | FFA25D FF629D FFE21D  |
       |                       |
       |  |<<     >>|    |>||  |
       | FF22DD FF02FD FFC23D  |
       |                       |
       |   -       +     EQ    |
       | FFE01F FFA857 FF906F  |
       |                       |
       |   0      100+   200+  |
       | FF6897 FF9867 FFB04F  |
       |                       |
       |   1       2      3    |
       | FF30CF FF18E7 FF7A85  |
       |                       |
       |   4       5      6    |
       | FF10EF FF38C7 FF5AA5  |
       |                       |
       |   7       8      9    |
       | FF42BD FF4AB5 FF52AD  |
       |                       |
       |          Car          |
```

```
      |          mp3          |
       \                     /
        ---------------------
      (FFFFFFFF for repeat when a button is held)


   2. In autonomous mode: See the accompanying ASMD chart noted on line 3.
      control path executes Moore statement in the states' state box, i.c. it turns
    the head in order to survey its surroundings for objects.
      Thereafter, the control path determines if there is an input signal
    corresponding to "exit"; if so, the control path clears the LCD subsystem and
    transitions back to the "Bootup" state.
      Otherwise, the control path continues to check whether there is lower than 10cm
     distance to an object; if true, system-of-interest moves backwards in order to
    increase the distance to object and checks the Mealy statement again.
      If the Mealy statement returns false, the control path continues to next Mealy
    statement in order to check whether the distance is greater than 10cm.
      In case that is true, the system-of-interest moves towards the object to
    decrease the distance and checks the Mealy statement again.
      Otherwise the control path loops back to the beginning of same state. \\


   3. In manual mode:
      The state box is empty so control path continues to first Mealy statement;
    checking whether a infrared signal for state-exit has been received.
      If not, control path continues towards second Mealy statement. In this
    statement, the control path determines whether a signal for moving forward has
    been received.
      If so, the system-of-interest moves forwards. Otherwise, the control path
    continues to check whether different input signals have been received, and in
    case they have, the corresponding action is executed.
      As seen in the ASMD chart, this continues until the control path reaches the
    last Mealy statement in "Manual\_serpentine" state.
      In case a signal for turning left is received, the system-of-interest will turn
     left, otherwise the control path loops back to the beginning of same state.
*/

// Including libraries:
#include <IRremote.h>                       // Library for the infrared
    communication with IR remote.
#include <Servo.h>                          // Library for servo motors.
#include <SM_iniCorrection.h>               // Library for correcting the servo motor
     at initialization.
// The rationale for an initial correction of the servo motors is:
// Whenever they are connected to power,
// either through battery or arduino,
// they make an initial turn/rotation of approx. 5-10 deg.

#include <Wire.h>                           // Library for I2C (LCD in our case)
    communication.
#include <LiquidCrystal_I2C.h>              // Library for I2C LCD functions (similar
     to LiquidCrystal library)



//--------------- IR related preamble ------------------------------------
// Defining the infrared receiving pin on Arduino(the datapin),
// instatiating a receiver at that pin and a variable that holds the value of what is
     received:

#define IR_DataPin 19                       // Infrared receiving pin on Arduino is
    defined
IRrecv IR_Receiver(IR_DataPin);             // An infrared receiver has now been tied
     to stated receiving pin
decode_results IR_Results;                  // A variable that holds the value of
    what has been received via IR communication.

// Declaring a constant IR value that will be used as the "exit" code depicted in
```

```
    first Mealy statements of "Auto" and "Manual_serpentine" states in ASMD chart.
// The value of this is FF52AD <--> key '9' on remote:
99
long int IR_Exit = 0xFF52AD;                 // A variable that holds an "exit"-hex
    value (key 9) for going back to BootUp State
101 long int IR_Forward = 0xFF18E7;           // A variable that holds an "move forward
    "-hex value (key 2)(for manual state)
    long int IR_Backward = 0xFF4AB5;          // A variable that holds an "move backward
    "-hex value (key 8)(for manual state)
103 long int IR_Right = 0xFF5AA5;            // A variable that holds an "move right"-
    hex value (key 6)(for manual state)
    long int IR_Left = 0xFF10EF;              // A variable that holds an "move left"-hex
     value (key 4)(for manual state)
105

107 //---------------- Servo motor related preamble ------------------------
// Creating servo objects, declaring how many and placing them in an array:
109 Servo Servo1;
    Servo Servo2;
111 Servo Servo3;
    Servo Servo4;
113
// Declaring how many servos are connected
115 const int N = 4;

117 // Create an array of all servo objects
    Servo servoArray[N] = {Servo1, Servo2, Servo3, Servo4};
119

121 // Declaring servo pins and putting them in an array
    int servo1Pin = 53;   //Head
123 int servo2Pin = 51;   //2nd link
    int servo3Pin = 49;   //3rd link
125 int servo4Pin = 47;   //4th link

127 int servoPinarray[N] = {servo1Pin, servo2Pin, servo3Pin, servo4Pin};  // Create an
    array of servo pins


129

131 // Declaring usual rotational values:
    int Origin_pos = 90;                      // Original position of servo motors
133 int Right_pos = 15 ;                       // 90-75 gives a 90 deg turn, link will be
    turning right
    int Left_pos = 165 ;                      // 90+75 gives a 90 deg turn , link will be
    turning right
135 int i = Origin_pos;                      // Global variable to turn the head and
    remember position


137
//---------------- DC Motor related preamble ------------------------------
139 // Defining input- and enable pins:
    #define EN 12  // white cable (used for speed control by adjusting PWM signal to
    motor)
141 #define IN2 7 // orange cable (-)
    #define IN1 6 // yellow cable (+)
143

145 //---------------- LCD related preamble ------------------------------------

147 // Instantiating a LCD screen and setting its address in one line:

149 LiquidCrystal_I2C LCD_Screen(0x27, 2, 1, 0, 4, 5, 6, 7);          // Format: (I2C_ADDR
    ,EN_PIN,RW_PIN,RS_PIN,D4_PIN,D5_PIN,D6_PIN,D7_PIN)
    // To find the address, see https://create.arduino.cc/projecthub/abdularbi17/how-to-
    scan-i2c-address-in-arduino-eaadda
151
```

```
153
//---------- Distance measuring sensor (HC-SR04) preamble ------- // For more
    info, see https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
155
// Defining pins for trigger (TRIG) and echo (ECHO):
157
#define tPin 13 // hvit
159 #define ePin 2 //gul

161 // Declaring variables for duration (how long it takes for a transmitted ultrasonic
    wave to be received)
// and distance (how far it is to an object)
163
unsigned long t;                                                    // Variable for the
    time a wave reflection registers.
165 // This is calculated by pulseIn(Eyes_echoPin, HIGH) function.
// Keep in mind that this calculates the time it takes for the wave to travel 2x the
    distance (forth and back)
167 // Unit: microseconds

169 int dist;                                                          // Variable for the
    distance(cm) to an object.
// The distance to an object is calculated by a manipulation of "Speed [m/s] =
    Distance [m] / Time [s]".
171 // In our case, the time given by pulseIn is for 2x distance traveled, which gives us
// --> 2x dist [m] = Speed [m/s] * t [s] <--> dist [m] = (Speed [m/s] * t [s]) / 2
173
// The speed of ultrasonic waves in air is approx. 340 m/s ~ 0.034 cm/microseconds (
    see https://www.ndk.com/en/sensor/ultrasonic/basic01.html)
175 // This gives a rationale for
// dist [cm] = (0.034 [cm/microseconds] * t [microseconds]) / 2;
177

179

181

183 //---------- State Machine related preamble ----------
// Declaring States of the State Machine. Our states, as depicted in ASMD chart, are
    {BootUp, Auto, Manual}.
185 // We will also need to declare to variables that can hold one of these states at a
    time:

187 enum State {BootUp, Auto, Manual};            // States declared by using enum (
    enumerated type)
State presentState, nextState;                  // A presentState and nextState
    variable of type "State" is used to hold what state we are in
189 // and which state to transition to next.

191

193 void setup() {

195   //Starting serial communication (UART, baudrate is 9600)):
   Serial.begin(9600);
197

199   //---------- State Machine related setup ----------
   // Initial State is Bootup:
201   presentState = BootUp;
   nextState = presentState;  // Default back to same state
203

205
   //---------- IR related setup ----------
207   // Initializing IR receiver for receiving process:
   IR_Receiver.enableIRIn();
209
```

```
  // Making the on-board LED on Arduino Mega 2560 blink during TX & RX (that is,
    during transmission):
211 IR_Receiver.blink13(true);

213 // Resetting the IR receiving value:
  IR_Results.value = 1;
215
  // Defining an interrupt so that whenever IR receiver gets new exit signal from
    remote, it is handled:
217 attachInterrupt(digitalPinToInterrupt(IR_DataPin), getIRval, CHANGE);

219 //------------- Servo motor related setup ------------------------------
  // Attaching all servos:
221 Servo1.attach(servo1Pin);
  Servo2.attach(servo2Pin);
223 Servo3.attach(servo3Pin);
  Servo4.attach(servo4Pin);
225
  // Initial correction of Servo motors:
227 // All connected servo motors are turned to their original position (facing
    straight)
  servoSetup(N, Left_pos);            // Then they're turned so that the link is
    facing left.
229 delay(1000);
  servoSetup(N, Right_pos);          // When we turn them back to the original
    position now, they're accurately positioned.
231 delay(1000);
  servoSetup(N, Origin_pos);
233

235 //------------- DC motor related setup ---------------------------------
  // Setting input- and enable pins as output:
237 pinMode(EN, OUTPUT);
  pinMode(IN1, OUTPUT);
239 pinMode(IN2, OUTPUT);

241


243 //------------- LCD related setup --------------------------------------
  // Start the LCD screen, turning the backlight on for brightness (visibility to
    read), and writing:
245 LCD_Screen.begin (16, 2); // 16 x 2 LCD module that we are using
  LCD_Screen.setBacklightPin(3, POSITIVE); // BL, BL_POL
247 LCD_Screen.setBacklight(HIGH);

249 // Display a nice welcome message for the user :-) (5 seconds) :
  LCD_Screen.setCursor(5, 0);         // Sets the cursor at 3rd column, 0th row to
    display "Hello :-)" in center
251 LCD_Screen.print("Hello!");         // Display message
  LCD_Screen.setCursor(2, 1);         // Sets the cursor at 3rd column, bottom row to
     display message centered
253 LCD_Screen.print("I'm Nagini :)"); // Display message
  delay(5000);                        // Hold it for 5 seconds
255 LCD_Screen.clear();                 // Clear screen

257


259
  //------------- Distance-measuring sensor (HC-SR04) setup ----------
261 // Setting the pinMode for the trigger to be an output, echo to be an input:
  pinMode(tPin, OUTPUT);
263 pinMode(ePin, INPUT);

265


267


269
```

```
271
}
273
void loop() {
275   // put your main code here, to run repeatedly:
   StateMachine();
277


279


281 }
283 void StateMachine() {
285   switch (presentState) {
        case BootUp:
287       BU();
          break;
289
        case Auto:
291       A();
          break;
293
        case Manual:
295       M();
          break;
297   }
}
299
void BU() {
301


303   // State box: Moore statement: Display following message on LCD: "Choose
        operational mode" on first row, "1(AUTO)/0(MAN)" on second row:
   LCD_Screen.home();                                          // Setting the
        cursor to 0,0
305   LCD_Screen.print("Choose operational mode:");              // Printing first
        message on top row

307   LCD_Screen.setCursor(0, 2);                               // Position cursor
        at 0th column, 2nd row
   LCD_Screen.print("1(AUTO) / 0(MAN)");                        // Printing second
        message on bottom row
309
   // First Mealy statement: Keep checking which state to transition to while there's
        no IR transmission. A 1 is Auto, 0 is Manual. 1 is IR code 0xFF30CF, 0 is IR code
        0xFF6897
311   // Clearing the variable for IR values first, so the following code doesn't use old
        values:
   //IR_Results = NULL;
313   //IR_Receiver.resume();


315
   if (IR_Receiver.decode(&IR_Results)) {                    // This will return TRUE if
        there's any results from IR transmission
317
     //Serial.println(IR_Results.value, HEX);                // For testing purposes: See the
        value in serial monitor
319
     switch (IR_Results.value) {
321     case 0xFF30CF:                                        // If the IR value is key '1' then
        the following will happen:
          LCD_Screen.clear();                                // Clear the LCD screen before
        displaying new message
323       LCD_Screen.home();                                  // Set cursor at 0,0
          LCD_Screen.print("Autonomous mode");               // Print "Autonomous mode selected"
        for 5 seconds
```

```
325         LCD_Screen.setCursor(0, 2);             // Position cursor at 0th column, 2nd
    row to finish message
            LCD_Screen.print("selected");
327
329         delay(4000);                            // Hold message for 4 seconds
            LCD_Screen.clear();                     // Clear screen
331
            //IR_Results.value = 0;                 // Reset the IR value
333
            IR_Receiver.resume();                   // Resume IR reception
335
337         nextState = Auto;                       // Set the state to transition to (Auto
     in this case)
339         break;
341       case 0xFF6897:                            // If the IR value is key '0' then the
    following will happen:
            LCD_Screen.clear();                     // Clear the LCD screen before
    displaying new message
343         LCD_Screen.home();                      // Set cursor at 0,0
            LCD_Screen.print("Manual mode");       // Print "Manual mode selected" for 5
    seconds
345         LCD_Screen.setCursor(0, 2);            // Position cursor at 0th column, 2nd
    row to finish message
            LCD_Screen.print("selected");
347         delay(5000);                            // Hold message for 5 seconds
            LCD_Screen.clear();                     // Clear screen
349
            IR_Receiver.resume();                  // Resume IR reception
351
            //IR_Results.value = 0;                // Reset the IR value
353
355         nextState = Manual;                     // Set the state to transition to (Manual
     in this case)
357         break;
359
            //          default:                          // If any other key is pressed,
    tell user to press 1 or 0
361         //            LCD_Screen.clear();             // Clear the LCD screen before
    displaying new message
            //            LCD_Screen.setCursor(0,0);      // Set cursor at 0,0
363         //            LCD_Screen.print("Please choose:");
            //            LCD_Screen.setCursor(0,1);
365         //            LCD_Screen.print("1(A) or 0(M)");
            //
367         //            LCD_Screen.setCursor(15,1);       // A timer to count down 5
    seconds so the user can choose again
            //            LCD_Screen.print("5");
369         //            delay(1000);
            //            LCD_Screen.setCursor(15,1);
371         //            LCD_Screen.print("4");
            //            delay(1000);
373         //            LCD_Screen.setCursor(15,1);
            //            LCD_Screen.print("3");
375         //            delay(1000);
            //            LCD_Screen.setCursor(15,1);
377         //            LCD_Screen.print("2");
            //            delay(1000);
379         //            LCD_Screen.setCursor(15,1);
            //            LCD_Screen.print("1");
381         //            delay(1000);
            //            LCD_Screen.setCursor(15,1);
```

```
383         //              LCD_Screen.print("0");
            //              //IR_Results.value = 0;
385         //
            //              break;
387
        }
389
        IR_Receiver.resume();  // Resume the IR reception process
391
    }
393
    presentState = nextState;  // Update the current state.
395
}
397
void A() {
399   ExitCheck();
    A_Mealy2();
401 }
403
405
407 void A_Scan() {
    if (i == Origin_pos) {  // If statement to check which direction to turn.
409   // If head is already positioned left, then it will turn right.
      // If it is already positioned right, then it will turn left.
411   // If it is in original position, then it will turn right.
      // Turning the head slowly from original position, to right:
413   for (i; i >= Right_pos + 25; i = i - 10) {
        Servo1.write(i);
415   }

417 } else if (Origin_pos < i <= Left_pos) {        // if the position of the head is
    left

419   // Turning the head slowly from left, to right:
      for (i; i >= Right_pos + 25; i = i - 10) {  // Turning the head from left to
    right
421     Servo1.write(i);
      }
423 } else if (Origin_pos > i >= Right_pos) { // if the position of the head is right
      for (i; i <= Left_pos + 25; i = i + 10) {  // Turning head from right to left
425     Servo1.write(i);
      }
427
    }
429
}
431
void DistPrint() {
433   LCD_Screen.setCursor(0, 0); // Sets the location at which subsequent text written
      to the LCD_Screen will be displayed
    LCD_Screen.print("Distance: "); // Prints string "Distance" on the LCD_Screen
435   LCD_Screen.setCursor(10, 0);
    LCD_Screen.print("   ");
437   LCD_Screen.setCursor(10, 0);
    LCD_Screen.print(dist); // Prints the distance value from the sensor
439   //LCD_Screen.clear();
    //  LCD_Screen.print("cm");
441   delay(250);
}
443
void DistCheck() {
445   digitalWrite(tPin, LOW);
    delayMicroseconds(2);
447
```

```
  digitalWrite(tPin, HIGH);
449  delayMicroseconds(10);

451  digitalWrite(tPin, LOW);

453  t = pulseIn(ePin, HIGH);

455  dist = t * 0.034 / 2;

457 }

459 void A_Mealy2() {
Second_Mealy:
461  DistCheck();
  DistPrint();
463
  if (dist < 10) {
465    // Move backwards:
    analogWrite(EN, 155);
467    digitalWrite(IN2, HIGH);
    digitalWrite(IN1, LOW);
469
    goto Second_Mealy;
471  } else {
    //    DistCheck();
473    //    DistPrint();
    A_Mealy3();
475  }
}
477
void A_Mealy3() {
479 Third_Mealy:
  DistCheck();
481  DistPrint();

483  if (dist >= 10) {

485    // Move forwards:
    analogWrite(EN, 255);
487    digitalWrite(IN2, LOW);
    digitalWrite(IN1, HIGH);
489
    goto Third_Mealy;
491
  } else {
493    //    DistCheck();
    //    DistPrint();
495    A_Scan();
    nextState = Auto;            // Start Auto again
497    presentState = nextState;    // Update state
  }
499 }

501 void M() {
  ExitCheck();
503
  if (IR_Results.value == IR_Forward) {  // If ir_signal is forward then move
    forwards
505    // Move forwards:
    analogWrite(EN, 255);
507    digitalWrite(IN2, LOW);
    digitalWrite(IN1, HIGH);
509  } else if (IR_Results.value == IR_Backward) {  // If ir_signal is backward then
    move backwards
    // Move backwards:
511    analogWrite(EN, 155);
    digitalWrite(IN2, HIGH);
513    digitalWrite(IN1, LOW);
```

```
  } else if (IR_Results.value == IR_Right) {  // If ir_signal is right then turn
    right
    // Turn head right:
    for (i; i >= Right_pos + 25; i = i - 10) {
      Servo1.write(i);
    }
    // Move forwards:
    analogWrite(EN, 255);
    digitalWrite(IN2, LOW);
    digitalWrite(IN1, HIGH);
  } else if (IR_Results.value == IR_Left) {   // If ir_signal is left then turn left
    // Turn head left:
    for (i; i <= Left_pos + 25; i = i + 10) {
      Servo1.write(i);
    }
    // Move forwards:
    analogWrite(EN, 255);
    digitalWrite(IN2, LOW);
    digitalWrite(IN1, HIGH);
  } else {
    nextState = Manual;            // Start Manual again
    presentState = nextState;    // Update state
  }


}




void ExitCheck() {
  // Check if the IR signal value is equal to the exit value:
  if (IR_Results.value == IR_Exit ) {    // In case it is a match, then the LCD_Screen
     is cleared, cursor is set to 0,0,
    // and a message saying "exiting" is displayed for 2 seconds.
    LCD_Screen.clear();
    LCD_Screen.home();
    LCD_Screen.print("Exiting...");

    // Correction of Servo motors:
    // All connected servo motors are turned to their original position (facing
    straight)
    servoSetup(N, Left_pos);                // Then they're turned so that the link is
     facing left.
    delay(1000);
    servoSetup(N, Right_pos);             // When we turn them back to the original
    position now, they're accurately positioned.
    delay(1000);
    servoSetup(N, Origin_pos);

    delay(2000);                          // Give the motors 2 seconds to adjust

    nextState = BootUp;                 // Transition back to Bootup if exit key is
    pressed

    presentState = nextState;

  }


}

void getIRval() {

  if (IR_Receiver.decode(&IR_Results)) {
    delay(200);
```

```
577         Serial.println(IR_Results.value, HEX);

579         IR_Receiver.resume();
        }
581 }
```