



Simon Fraser University,  
8888 University Dr.  
Burnaby, BC Canada  
Email: rmt3@sfu.ca

*November 8, 2012*

Dr. Andrew Rawicz  
School of Engineering Science  
Simon Fraser University  
Burnaby, British Columbia  
V5A 1S6

**Re: ENSC 440 Design Specification for a QuickScan Mapping Device**

Dear Dr. Rawicz,

Enclosed is our *Design Specification Document for a Quick Scan Mapping Device*, which further describes our company's product. We are designing a portable scanning device that will quickly measure the area of a given room in 3D space. This device will eliminate the need of manually measuring dimensions using a measuring tape or similar tools.

The design specification describes our implementation process of high-level requirements. These requirements were priority *PA* specifications (i.e. high priority) in the functional specification document. We will use this design document as a guide for research and development activities. As well, we will use it for unit and integration testing as the minimum standard of requirement.

Dimension Technologies consists of five senior engineering students: Chris Kwong, Oliver Huang, William Chiang, Rahul Thomas, Jack Zhang. We have a wide range of abilities with systems, electrical, and computing background.

If there are any questions or concerns regarding our document or device, please contact me by phone at 604-807-0486 or by e-mail at rmt3@sfu.ca.

Sincerely,

A handwritten signature in black ink, appearing to read "Rahul Thomas", with a stylized flourish at the end.

**Rahul Thomas**  
CEO  
Dimension Technologies

**Enclosure: *Design Specification for a QuickScan Mapping Device***



**DESIGN SPECIFICATION:**  
QUICKSCAN MAPPING DEVICE

Rahul  
Thomas

CEO

Chris Kwong

CTO

Yumin Oliver  
Huang

CFO

Jack Zhang

COO

William  
Chiang

Managing  
Director

**Contact Person:**

Rahul Thomas  
rmt3@sfu.ca

**Submitted To :**

Dr. Andrew Rawicz-ENSC 440  
Steve Whitmore-ENSC 305

School of Engineering Science  
Simon Fraser University

**Issued Date:**

November 8, 2012



## Executive Summary

The QuickScan Mapping Device is a product that Dimension Technologies is developing to assist building professionals in the industrial & residential sectors. The present rapid growth in construction and real-estate that the world is experiencing has led to an increase in the demand for facilities and houses. Typically, one main consideration in each of these cases is the size and dimensions of different property areas. Dimension Technologies' product will solve these questions through a relatively low cost product. While other solutions have been developed in the past, QuickScan is a standalone device that measures room dimensions in a very short time.

This document will outline the design process and implementation process for critical functional requirements. This document will begin by summarizing the major components of our system. Given the complexity of the entire device, each component has been subdivided into smaller sections to increase the document's readability. For each subsystem, our design considerations have been well documented indicating how we plan to implement each specification in detail.

The QuickScan module consists of a high precision depth sensor that sends data through various signal processing hardware and tools in order to create a 3D render of a given room or area onto the controlling laptop. There are many features and functions that we would like to support in order to best meet our customers' needs. However, in order maintain a tight schedule and budget; we have sorted out these desired features into different levels of priority. This document will explain how we will implement high priority features.

Some of the subsystems that will be discussed in detail are:

- Mechanical Subsystem - Ranging from electromechanical devices to device enclosures
- Electrical Subsystem - Includes power and control circuitry
- Software Subsystem - Consists of data acquisition, data processing and user interaction

Additionally, a test plan has been included in this document. The QuickScan device and all its subcomponents will be thoroughly tested using the unit tests mentioned in this document.

Our proposed design features will be implemented in the final QuickScan module. The targeted completion date is the first week of December 2012.



# Table of Contents

<b>1.</b>	Introduction .....	1
1.1	Scope .....	1
1.2	Intended Audience.....	1
<b>2.</b>	System Overview.....	2
<b>3.</b>	Hardware .....	5
3.1	Microsoft Kinect .....	5
3.2	Arduino Nano .....	9
3.3	Stepper Motor - ROB-09238.....	10
3.4	ROB-10267 Motor Driver .....	13
3.5	Honeywell HMC5883L Triple Axis Compass .....	17
3.6	Other Components .....	19
<b>4.</b>	Mechanical .....	20
<b>5.</b>	Software Processing .....	26
5.1	Data Processing.....	26
5.2	Software Processing .....	28
5.3	QuickScan Procedure.....	30
5.4	Graphical User Interface (GUI) .....	32
<b>6.</b>	Arduino Software.....	34
6.1	Serial Communication Protocol.....	34

6.2 Arduino Microcontroller Logic .....	38
7. Electrical .....	41
7.1 Circuit Design & Topology.....	41
7.2 Battery Powering Considerations .....	41
8. Test Plan .....	45
9. Environmental Considerations.....	49
9.1 Project Disposal .....	49
9.2 Product Disposal .....	49
10. Conclusion.....	50
11. References.....	51

# List of Figures

Figure 2.1 System Overview.....	2
Figure 2.2 QuickScan Mapping Device [28][29] .....	4
Figure 3.1.1 Kinect Sensors [14].....	6
Figure 3.1.2 PrimeSense [12].....	6
Figure 3.1.3 Kinect Features .....	7
Figure 3.1.4 Kinect IR coding Image .....	8
Figure 3.1.5 Depth map (left) created from light IR pattern (right).....	8
Figure 3.2.1 Arduino Nano [3].....	9
Figure 3.3.1 ROB-09238 Stepper Motor [4].....	11
Figure 3.3.2 Stepper Motor Overview [22].....	11
Figure 3.3.3 Two Phase [22] .....	12
Figure 3.4.1 Motor Driver [5] .....	13
Figure 3.4.2 Phase input with Micro-stepping [20] .....	14
Figure 3.4.3 Phase input with Micro-stepping [20] .....	15
Figure 3.4.5 [2][3][5] .....	16
Figure 3.5.1 HMC5883L Triple Axis Compass [6] .....	17
Figure 3.5.2 I <sup>2</sup> C Overview [18] .....	18
Figure 4.1 Overview of the QuickScan device [28][29] .....	20



Figure 4.2 QuickScan Diagram [29]	21
Figure 4.3 Operation of the two stepper motors	22
Figure 4.4 Pan-Tilt Enclosure [4]	23
Figure 4.5 Pan-Tilt Assembly Diagram [4]	24
Figure 4.6 Motor Mechanical Diagram [4]	25
Figure 5.1.1 Forward Kinematics	26
Figure 5.1.2 Sample stitch of two snapshots	27
Figure 5.2.1 MeshLab Output	29
Figure 5.3.1 QuickScan Procedure	30
Figure 5.3.2 Illustration of QuickScan Process	32
Figure 5.4.1 QuickScan GUI	33
Figure 6.1.1 Arduino & Processing Synchronization	35
Figure 6.1.2 Stepper Motor 0 Protocol	36
Figure 6.1.3 Magnetic Bearing Protocol	36
Figure 6.1.4 Stepper Motor 1 Protocol	37
Figure 6.2.1 Arduino Logic	38
Figure 6.2.2 Motor Control Logic	39
Figure 6.2.3 Motor Interrupt Subroutine Logic	40
Figure 7.1.1 Circuit Schematic	41
Figure 7.1.2 Wiring Diagram	42
Figure 7.2.1 QuickScan Battery	43

# List of Tables

Table 3.2.1 Arduino Nano [3][16] .....	10
Table 3.3.1 Stepper Motor Specs [4] .....	12
Table 7.2.1 Component Voltage & Current .....	43

# Glossary

**3D** – Three Dimensional  
**ADC** – Analog to Digital Converter  
**CAD** – Computer Aided Design  
**CMOS** – Complementary Metal Oxide Semiconductor  
**COM** – Communication  
**CPU** – Central Processing Unit  
**DC** – Direct Current  
**GUI** – Graphical User Interface  
**I/O** – Input Output  
**I<sup>2</sup>C** – Inter-Integrated Circuit  
**IC** – Integrated Circuit  
**ID** – Identification  
**IDE** – Integrated Development Environment  
**IR** – Infrared  
**Li-ion** – Lithium Ion  
**PCB** – Printed Circuit Board  
**PIC** – Peripheral Interface Controller  
**PWM** – Pulse Width Modulation  
**PWR** – Power  
**RGB** – Red Green Blue  
**SDA** – Serial Data line  
**SCL** – Serial Clock  
**TTL** – Transistor-Transistor Logic  
**UART** – Universal Asynchronous Receiver/Transmitter  
**USB** – Universal Serial Bus  
**VGA** – Video Graphics Array

# 1. Introduction

The QuickScan mapping device is a relatively low cost device that can be used by drafters and realtors for various dimensioning applications. It will enable professionals in this area to quickly make measurements of a given room and create a viewable file in industry standard CAD software such as MeshLab and AutoCAD. The device will be relatively low cost and will provide quick and accurate measurements of a desired room. Our objective is to save professionals time and money in the measurement and floor planning process. Our design considerations and process are outlined in this document.

## 1.1 Scope

This document will outline in detail how we plan to implement the requirements mentioned in the functional specification document. Please note, we will focus on high and moderate priority requirements and their implementations in this document.

## 1.2 Intended Audience

This document will primarily be used by Dimension Technologies' team in the final implementation of our product. Also, it will serve as a functional and design benchmark for our working device. Specifically, the test cases mentioned will be used in the testing stage of our development process.

## 2. System Overview

A high level overview of the QuickScan device can be seen in Figure 2.1. The diagram contains the major hardware and software systems that have been weaved together to create the end product. Please note that each of these subsystems will be explained in much detail in the following sections.

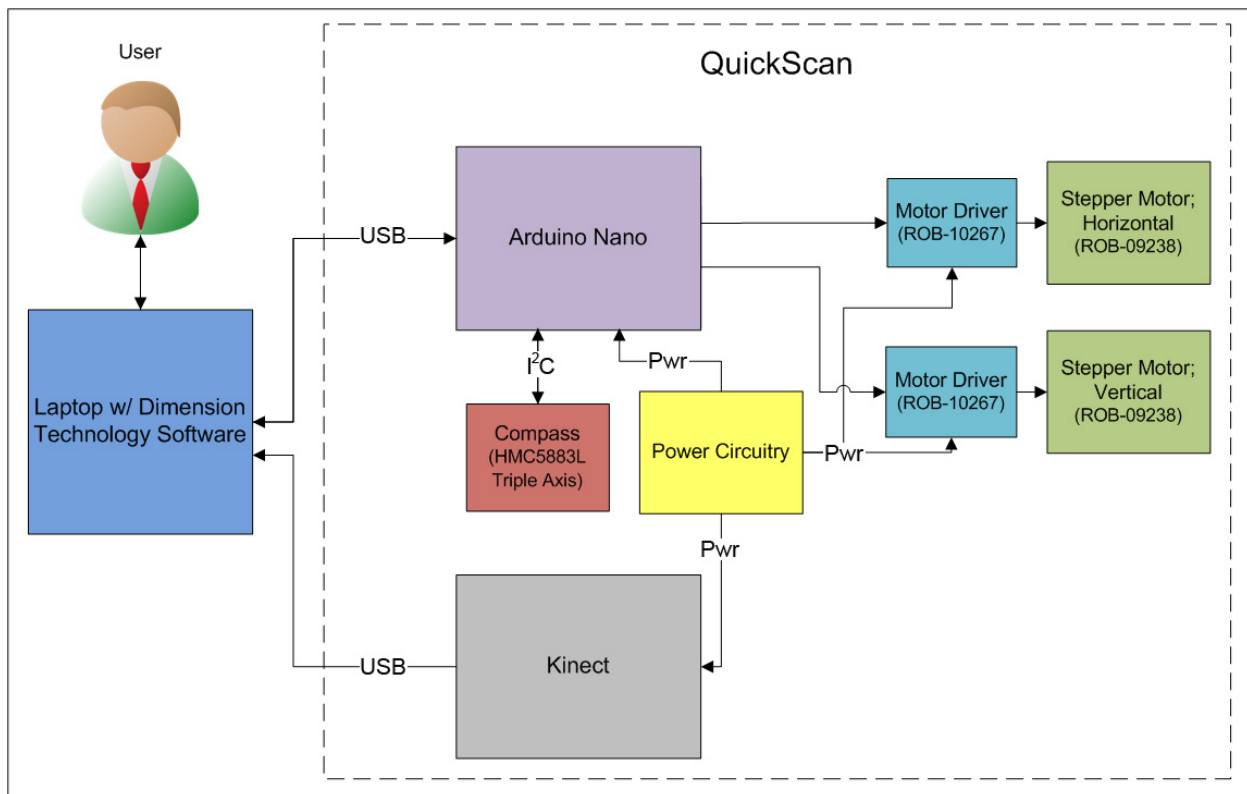


Figure 2.1 System Overview

Inherently, our overall system can be divided into functioning components. The first component is the QuickScan device itself which incorporates the depth sensing and other electromechanical components. The other component is the user's laptop which will be running Dimension Technologies' software to enable interfacing with the main QuickScan device; ultimately producing a viewable CAD drawing.

The user's sole method of interaction will be through the GUI that will be installed on the user's laptop. This laptop will have the needed QuickScan software installed. A USB connection will be used to interface with the QuickScan device. All control parameters and other options will be set from the GUI on the laptop. One of the parameters will be a list of COM ports the user can select to connect to the QuickScan device. The QuickScan module can be connected to a desired laptop through USB.

Inherent to this system is the high precision depth and RGB sensing system. We will use a Kinect depth sensor with an integrated RGB sensor. The data that is given off by this sensor is raw point cloud data which is then processed to create a render of the room. More specifications of this device are given in the next section. The Arduino microcontroller will serve as a medium between the laptop and other hardware using serial communication via the USB.

The device will consist of two integrated motors which will enable a wide range of data collection. The first motor is used to enable a full 360° horizontal range of acquisition. The second motor is used for a vertical tilt to enable a greater vertical acquisition range. This motor has a range of about 70° in both vertical directions.

QuickScan also features a battery powered capability. This is to enable maximum flexibility in positioning the entire device in an area. The hassle of wires and other cables are eliminated. This battery powering system consists of a rechargeable 12V battery with an on-device regulator.

Figure 2.2 displays a 3D model of our device showing the major components.



Figure 2.2 QuickScan Mapping Device [28][29]

## 3. Hardware

This section will identify the different hardware components that will be used in our product. These components have been manufactured by third party sources and will be integrated into our system. Our project utilizes the following hardware components:

- Microsoft Kinect
- Arduino Nano
- Stepper Motor (ROB-09238)
- EasyDriver Stepper Motor Driver
- Triple Axis Digital Compass IC HMC5883L by Honeywell

The following sections will give an overview of each component, a brief technical description, and how it integrates into QuickScan.

### 3.1 Microsoft Kinect

The Kinect [1] [2] is a motion sensing device that has been developed by Microsoft. It features 3D depth sensors and an RGB camera. Its initial intended use was as a peripheral to Microsoft's XBOX gaming console. However, due to its unique combination of sensors, the Kinect can be use in a variety of other applications. Figure 3.1.1 shows an image of the Kinect outlining its major features:



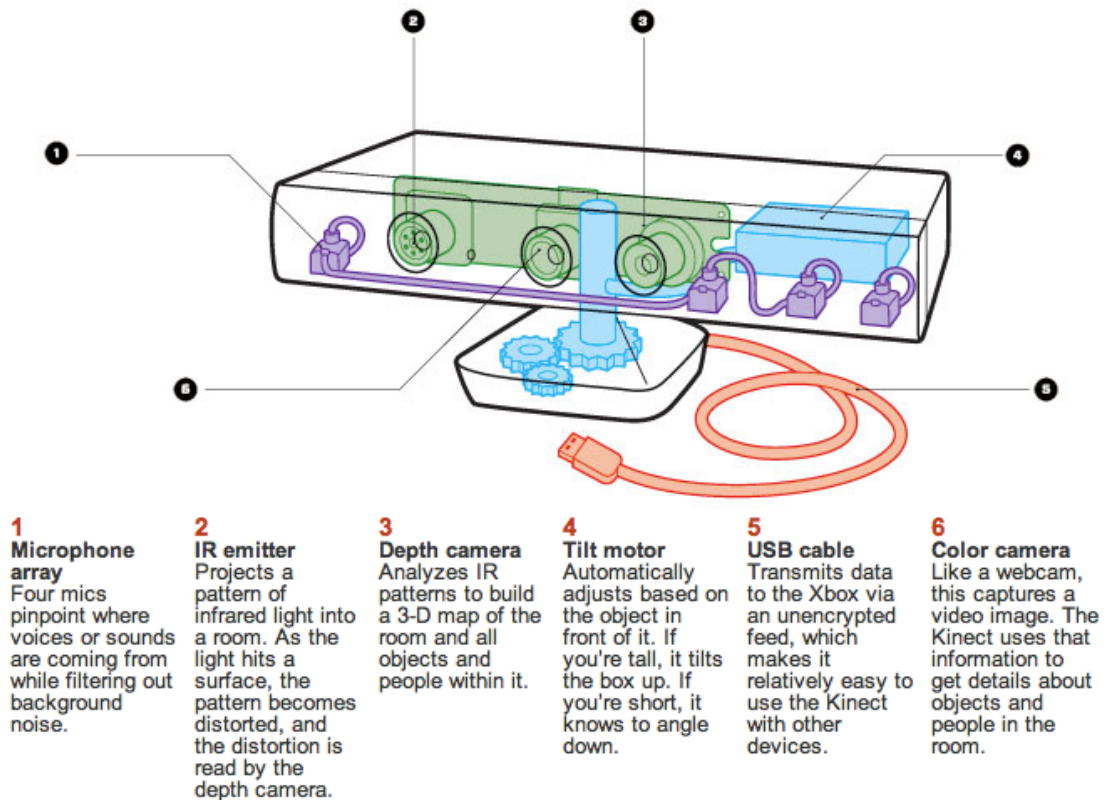


Figure 3.1.1 Kinect Sensors [14]

The Kinect's depth sensing features were created by an Israel-based company called PrimeSense [13]. The PrimeSense device with relation to the Kinect is shown in the follow figure:

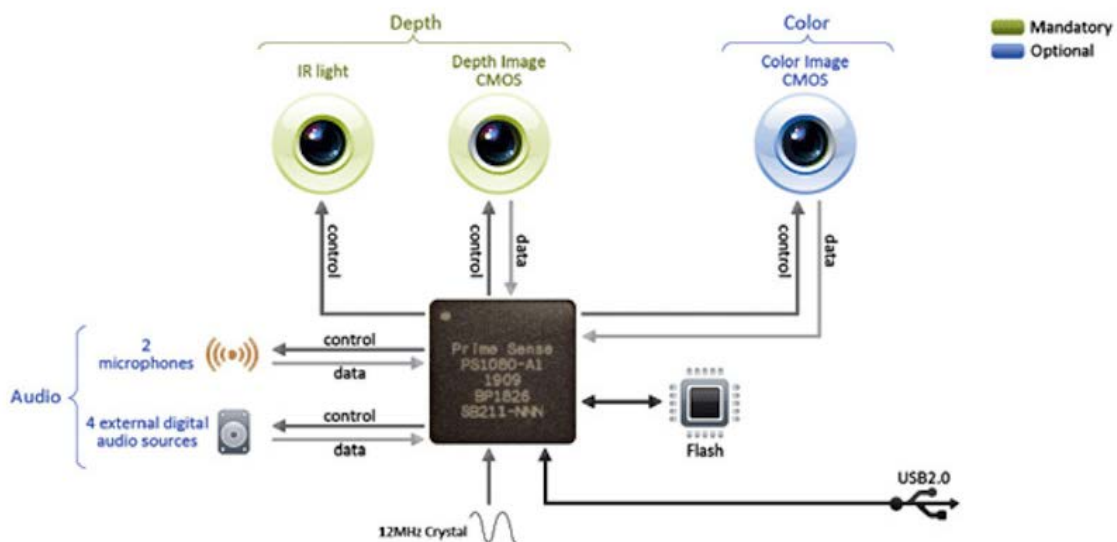


Figure 3.1.2 PrimeSense [12]

There are 3 main features in Figure 3.1.2: [13]

- Color image sensing
- IR light source – combination of laser diode and diffuser
- IR image source – used for depth measurements

In essence, the Kinect measures distances by using IR to measure depths using the depth sensor. The data that is given off by the Kinect is called point cloud data. Point cloud data is a group of x, y, and z coordinates (relative to the Kinect sensor) that tell the location of every infinitesimal small region of the object in front of the sensor. Each small region is assigned an x, y and z coordinate. In addition, each region is assigned an RGB value for color identification.

The Kinect operates on a 12V power supply and can be connected to using a USB connection.

#### Microsoft Kinect:

To create a 3D point cloud of a physical room, the QuickScan uses innovative design of the Kinect sensor developed by Microsoft. The Kinect sensor combines several technologies to effectively collect 3D information. As shown in Figure 3.1.4, the Kinect Sensor features an 8-bit VGA resolution (640x480) camera, infrared (IR) projector and an infrared(IR) Camera.

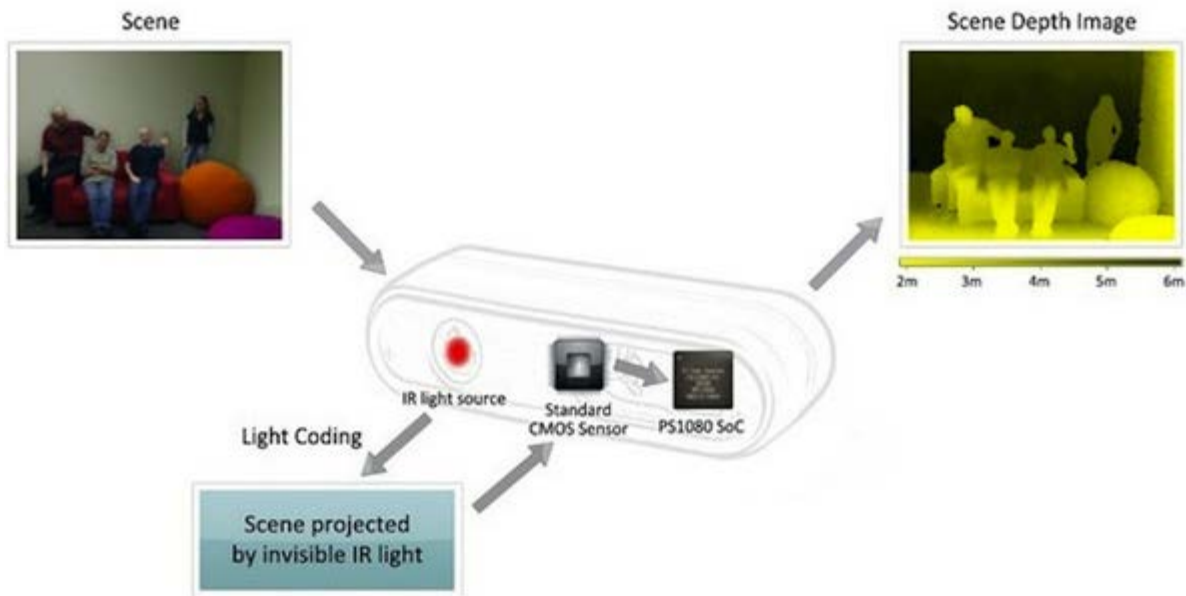


Figure 3.1.3 Kinect Features [14]

A virtual depth camera is formed with the combination of the IR projector and the IR Camera when their data streams are synergized with an internal chip developed by PrimeSense allowing the Kinect Sensor to create a 3D depth field of the scene in front of the Kinect.

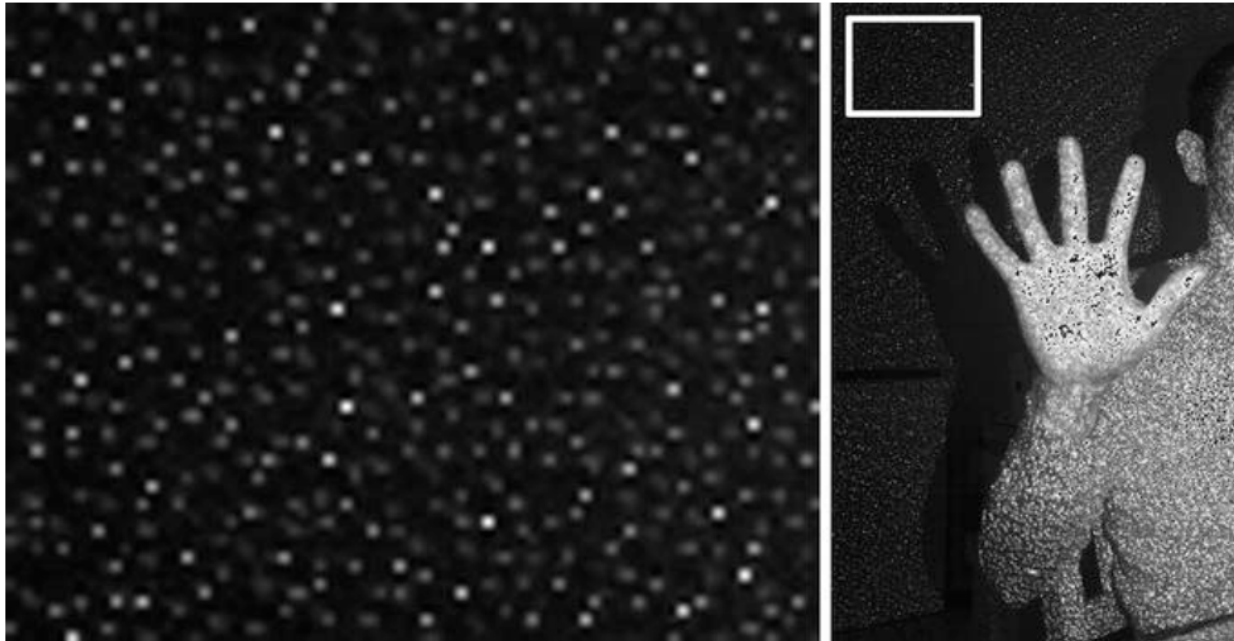


Figure 3.1.4 Kinect IR coding Image [30]

The IR project beams a pattern of infrared light beams (IR coding image) into the environment in front of the sensor. The IR beams are then reflected on objects and is captured by the CMOS image sensor. This data is relayed to the PrimeSense chip and is translated into a depth image as seen in Figure 3.1.5.

When the depth camera combined with the RGB camera, the Kinect sensor can 3D motion caption the environment in front of the Kinect at 30 frames per second.



Figure 3.1.5 Depth map (left) created from light IR pattern (right) [30]

Figure 3.1.5 demonstrates, the Kinect can create a depth maps based on the light infrared pattern captured by the CMOS IR Camera.

The drivers and interface (OpenNI) used to communicate with the intricate sensors on other Kinect is explained in more detail in the section 5.2.

#### Application:

In our product we will use the Kinect for its depth sensing and RGB camera. The Kinect features a very high precision depth sensor that can detect objects up to 10m away. Its field of view is 57° horizontally and 43° vertically. Due to its limited acceptance angle, a rotation system is needed to increase the range of acquisition (more on this later). Our floor plans will be created using the depth information that is given by the Kinect.

## 3.2 Arduino Nano

The Arduino Nano features the Atmega328 microcontroller and other necessary components fitted into a small compact PCB board as seen in Figure 3.2.1:

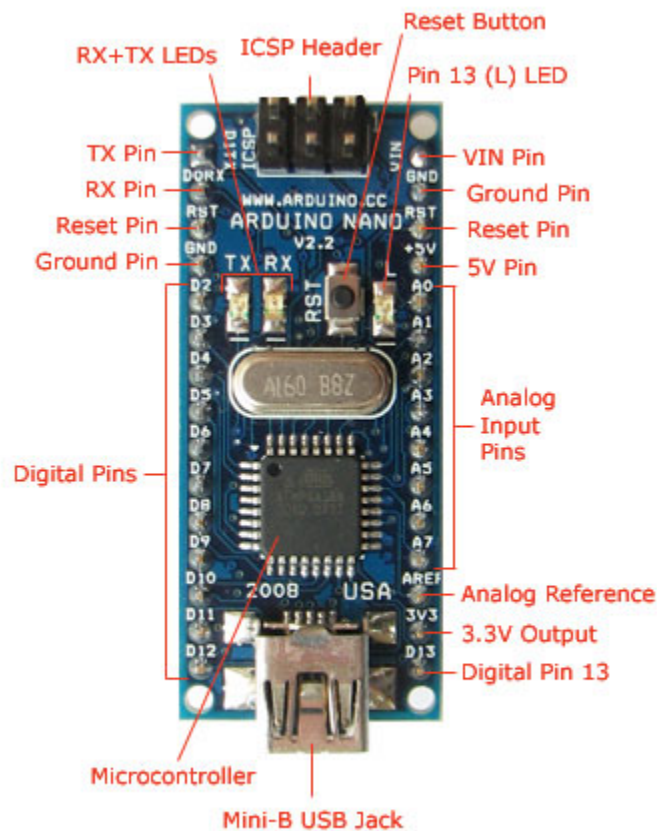


Figure 3.2.1 Arduino Nano [3]

### Overview [15]:

The Arduino utilizes its own IDE and programming language based on C++. This makes the Arduino much easier to program compared to other microprocessors (i.e. PICs), though still highly flexible as users can create and integrate their own libraries developed in C++.

The open-source nature of this device means a vast resource available for developers. It is well supported through online open-source discussion communities which provide documentation and Arduino libraries keeping developing time low. In addition, the modular integration of many hardware components makes the Arduino a very powerful platform and the most suitable logic unit for our project. The specifications of the Nano are shown in Table 3.2.1:

**Table 3.2.1 Arduino Nano [3][16]**

Specification	
Operating Voltage	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14 (6 provide PWM output)
Analog Input Pins	8
DC Current per I/O Pin	40 mA
Flash Memory	32 KB
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz
Dimensions	0.73" x 1.70"

### Application:

The slim form factor of the Arduino Nano combined with its ease of use and serial communication makes this the optimal component for relaying data between the computer and the other hardware

## 3.3 Stepper Motor - ROB-09238

As mentioned in the system overview, due to limitations in the Kinect's acquisition range we needed to implement a system that would enable us to reach maximum coverage. This can be solved using stepper motors. The stepper motor design was chosen due to its high level of accuracy and relatively low cost. The QuickScan device makes use of two ROB-09238 stepper motors [4]; seen in Figure 3.3.1:



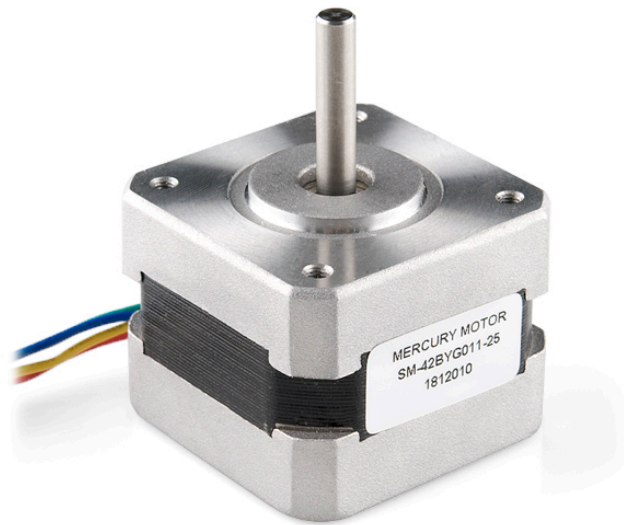


Figure 3.3.1 ROB-09238 Stepper Motor [4]

**Technical Overview:**

Stepper motors are simple, but powerful for applications where accurate mechanical rotation is needed. Unlike the conventional DC motor, stepper motors feature a brushless design, controlled by phases, electromagnets (stator poles) placed around the rotary gear in the middle. Through sequential activation of these phases, the stepper motor can be commanded to move and hold at a specified location.

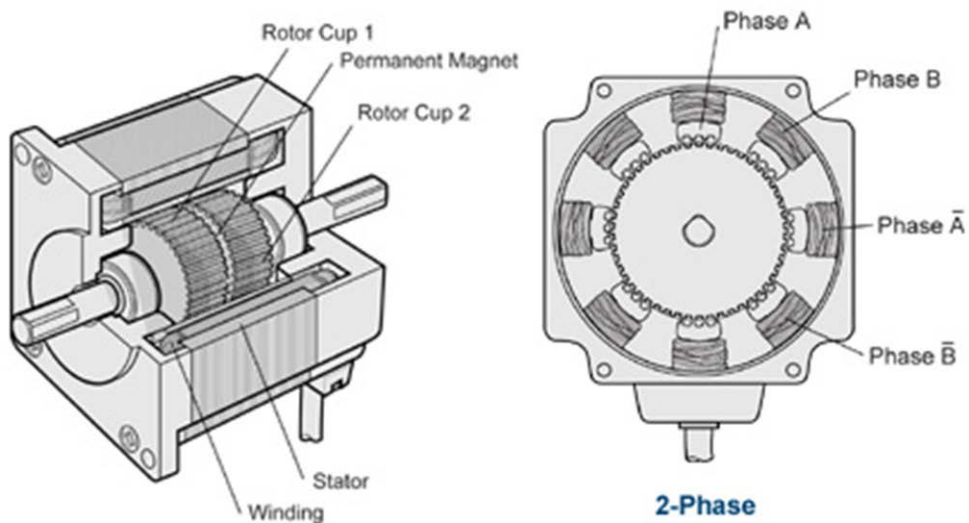


Figure 3.3.2 Stepper Motor Overview [22]

The ROB-09238 stepper motor is a 4 wire 2 phase unipolar stepper motor capable of turning 1.8° per step and accurate to 0.09°. Below are more detailed properties for the ROB-09238 motor:

Table 3.3.1 Stepper Motor Specs [4]

Specification	
Step Angle	1.8 °
Max Step Error	0.09 °
Phase	2
Rated Voltage	12 V
Rated Current	333 mA
Holding Torque	2.3kg*cm
Drive Shaft Diameter	5 mm
Winding Resistance	32.6 Ω
Winding Inductance	48 mH
Max Flux Linkage	1.8 Vs
Max Detent Torque	0.016 N.M
Total inertia	3.5 Kg.m.m
Total friction	4 Kg.m/s

Based on a stepper motor design, stepper motors have a defined step describing the minimum turn the motor can make. Every revolution is divided into a discrete number of steps, in this case of the ROB-09238, 200 steps ( $360^{\circ}/200^{\circ} = 1.8^{\circ}$  per step). To rotate the stepper motor to a specific angle, the user will calculate the number of steps it will take and manipulate the control wires to activate the two phases accordingly as seen below.

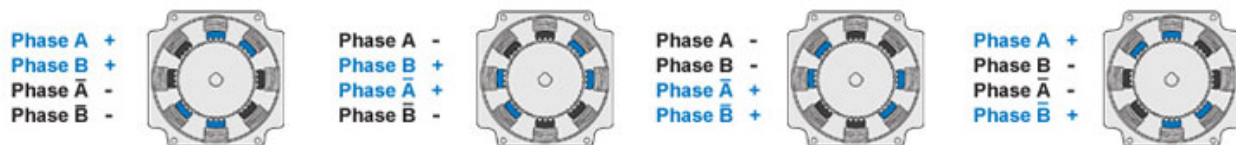


Figure 3.3.3 Two Phase [22]

**Application:**

As a result of how the stepper motors operate, they are very good at turning to a specific point accurately, at user controlled speed and direction, which are several good reasons why the decision was made to use stepper motors instead of the alternative: servo motors.

Though servo motors are comparatively easier to control and more efficient, its ability to only turn 180° proved to be too limiting for the QuickScan. In addition, servo motor requires feedback from an encoder to determine its position which can have slight variations between motor to motor making their accuracy a big issue.

With more holding torque, flexible control and continuous rotation, the QuickScan features 2 ROB-09238 to control its vertical and horizontal rotation.

### 3.4 ROB-10267 Motor Driver

In order to increase the flexibility of our design we chose to include motor drivers for each stepper motor mentioned above. The motor driver that was selected is the ROB-10267 EasyDriver Stepper Motor Driver [5]. The stepper motor driver can be seen in Figure 3.4.1.

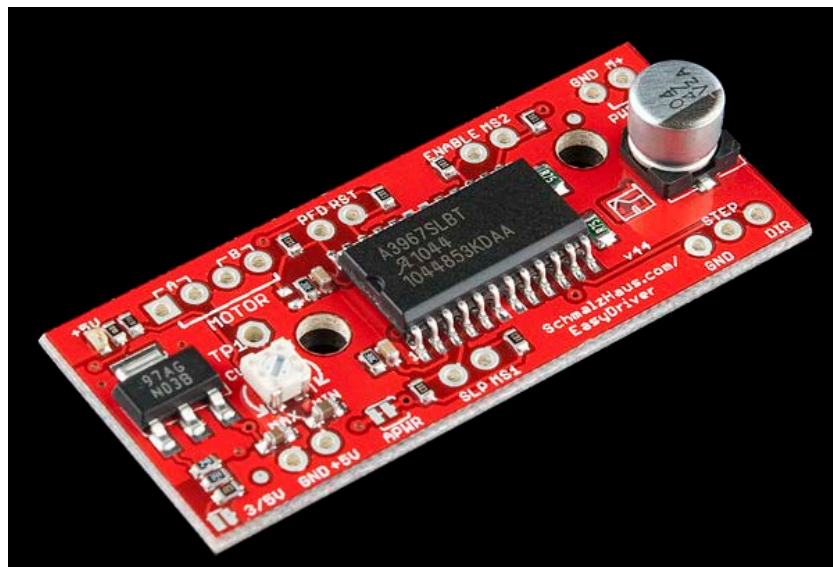


Figure 3.4.1 Motor Driver [5]

#### Technical Overview [5]:

Some key specifications for this stepper driver are [5]:

- Power supply range from 7 – 30 V
- Compatible with 4, 6, and 8 wire stepper motors of any voltage
- Adjustable current control from 150mA/phase to 750mA/phase
- Enables micro-stepping resolution to full, half, quarter and eighth steps

The EasyDriver features the A3967SLB stepper motor driver chip allowing the microcontrollers such as the Arduino [23][24] to easily control 2 phase stepper motors. The EasyDriver removes the complexity of correctly activating phases for each step and instead takes pulses into the STEP pin to automatically increment its step. The user can easily adjust the stepper motor speed by varying frequency of these pulses.



In addition the driver includes a micro-stepping motor feature with built-in translator where the user can specify full, half-, quarter, and eighth-step modes which will be explained in more detail in the next section.

The EasyDriver requires external power to drive the stepper motor which is satisfied by a 12V lithium ion battery. The A3967SLB also includes a current regulator that has the ability to operate in slow, fast, or mixed current-decay modes. This current-decay control scheme results in reduced audible motor noise, increased step accuracy, and reduced power dissipation.

**Micro-Stepping:**

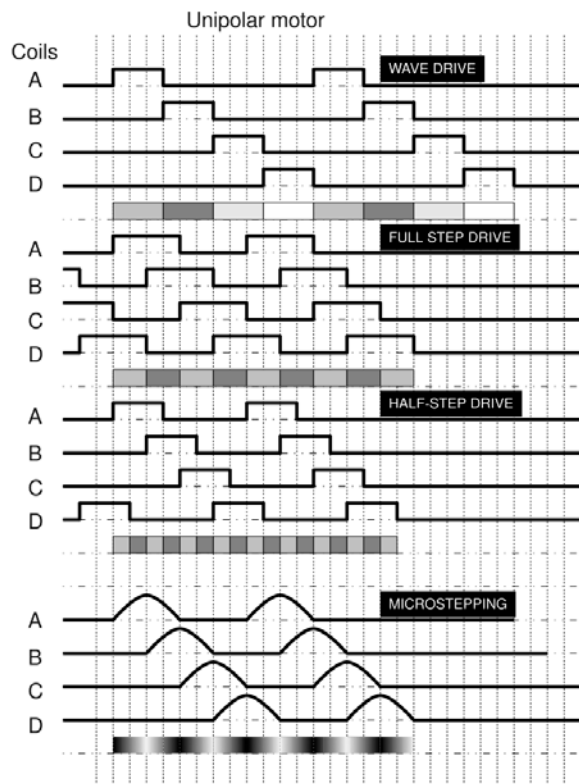


Figure 3.4.2 Phase input with Micro-stepping [20]

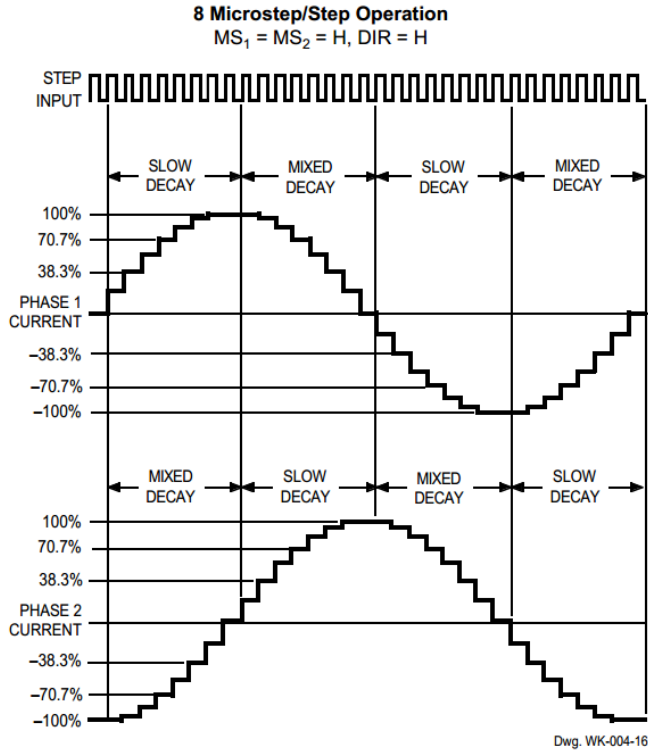


Figure 3.4.3 Phase input with Micro-stepping [20]

By micro-stepping the stepper motor, the user can divide the step angle further with half stepping, quarter stepping and in the QuickScan application, eighth-stepping effectively dividing the step size from 1.8° as you can see in Figure 3.4.2.

As seen in Figure 3.4.3, by decreasing the current to one phase while increasing the current to the next phase in increments, the user can increase the step resolution 8 times using the EasyDriver's default "8 Micro-step" operation. In addition to providing a higher resolution, micro-stepping also ensures smoother operation and less vibration.



Applications:

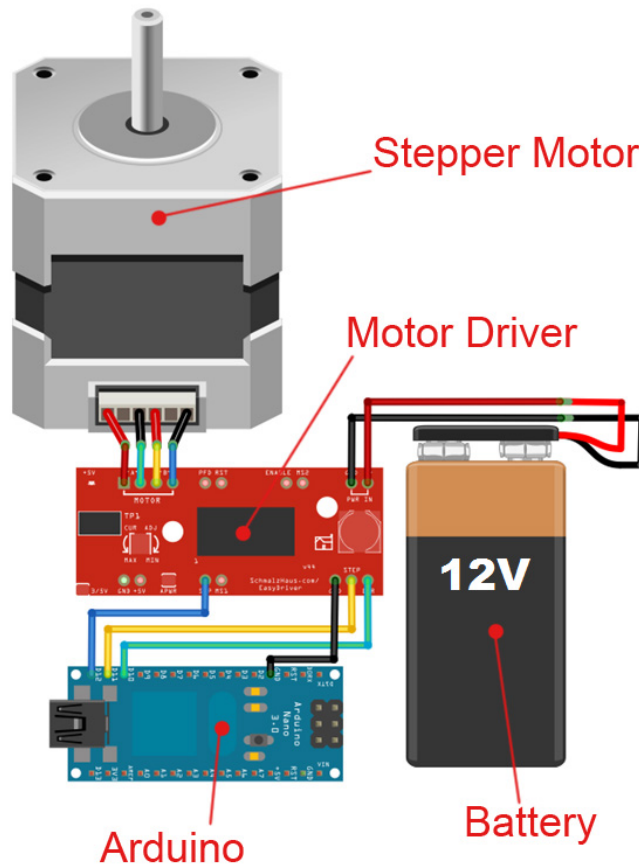


Figure 3.4.5 [2][3][5]

As seen in the diagram above, the 4 wires from the stepper motor controls 2 phases, where the red and green wires represent one phase and the yellow and blue wire presents the other. The stepper motor and EasyDriver is powered by the external 12V battery, while the Arduino Nano receives power directly from the USB connection.

It can also be observed that the Arduino Nano uses three control pins to send commands to the EasyDriver:

- Blue – (SLEEP) Enables or disables the EasyDriver and Stepper Motor
- Yellow – (STEP) Step input from the Arduino
- GREEN – (DIR) Determines the rotation direction of the Stepper Motor



### 3.5 Honeywell HMC5883L Triple Axis Compass

One of the applications that Dimension Technologies is targeting is the drafting industry; specifically to help in the floor planning and blueprint creation process. In any floor plan one key element is the identification of cardinal coordinates (North, East, West, and South). There were many options that were available to us. We chose to stick with a relatively simple option: Digital Compass. In particular, we chose the Honeywell HMC5883L Triple Axis Compass Module [6]. An image of this compass is shown in Figure 3.5.1:

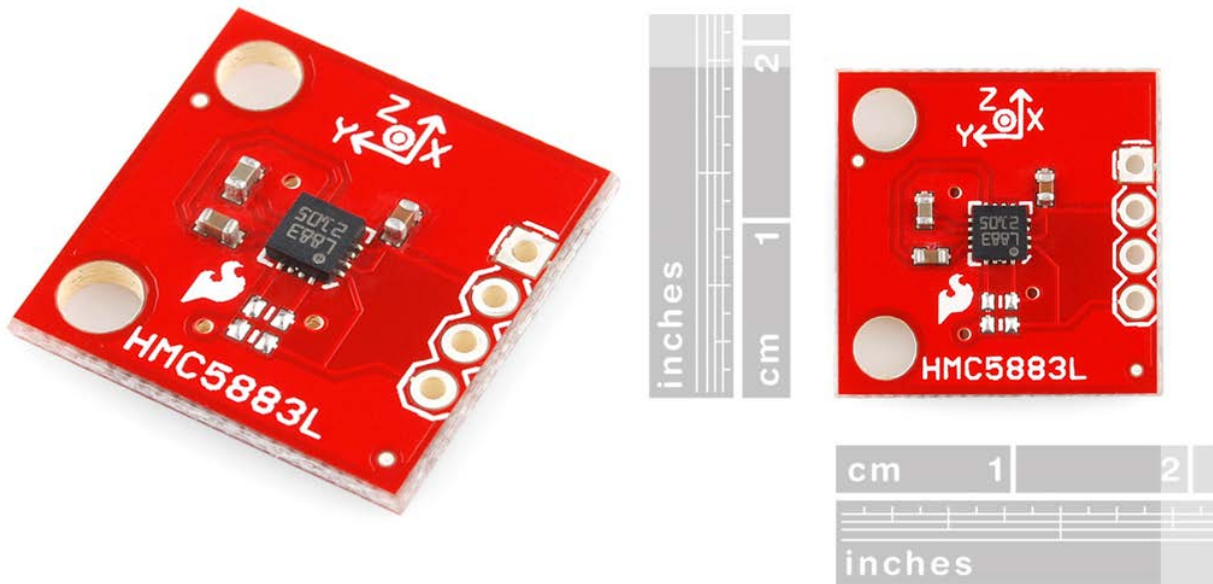


Figure 3.5.1 HMC5883L Triple Axis Compass [6]

#### Technical Overview:

Some of the HMC5883L's key features include [6][25]:

- I<sup>2</sup>C interface
- 2.16-3.6V DC voltage supply
- Current draw: 100 $\mu$ A (measurement mode), 2 $\mu$ A (idle mode)
- 5 milli-gauss resolution

The HMC5883L is a three axis magnetometer capable of sensing its surrounding magnetic field in 3 directions. A normal mechanical compass [27] works by aligning itself to the earth's magnetic field, however instead of using any mechanical parts, the HMC5883L uses three magneto-resistive sensors on three different axis. The current flow through the sensor varies depending on the

orientation of the sensor to the earth's magnetic field. The internal circuitry will automatically store the data inside its registers.

The HMC5883L's functionality includes automatic degaussing strap drivers, offset cancellation and a 12-bit ADC enabling 1° to 2° compass heading accuracy. The HMC588L operates at a low operational voltage range of 2.16V to 3.6V at 100 $\mu$ A.

### I<sup>2</sup>C Protocol Overview [17][18][26]:

The Arduino interfaces with the HMC5883L with the I<sup>2</sup>C protocol and configured such that the Arduino Nano is the master and the HMC5883L is the slave.

The I<sup>2</sup>C protocol features shared data (SDA) and clock (SCL) bi-direction lines between all I<sup>2</sup>C devices on the bus. The SDA line communicates important data between the master and slave devices and the SCL clock line is used to synchronize the data transfer.

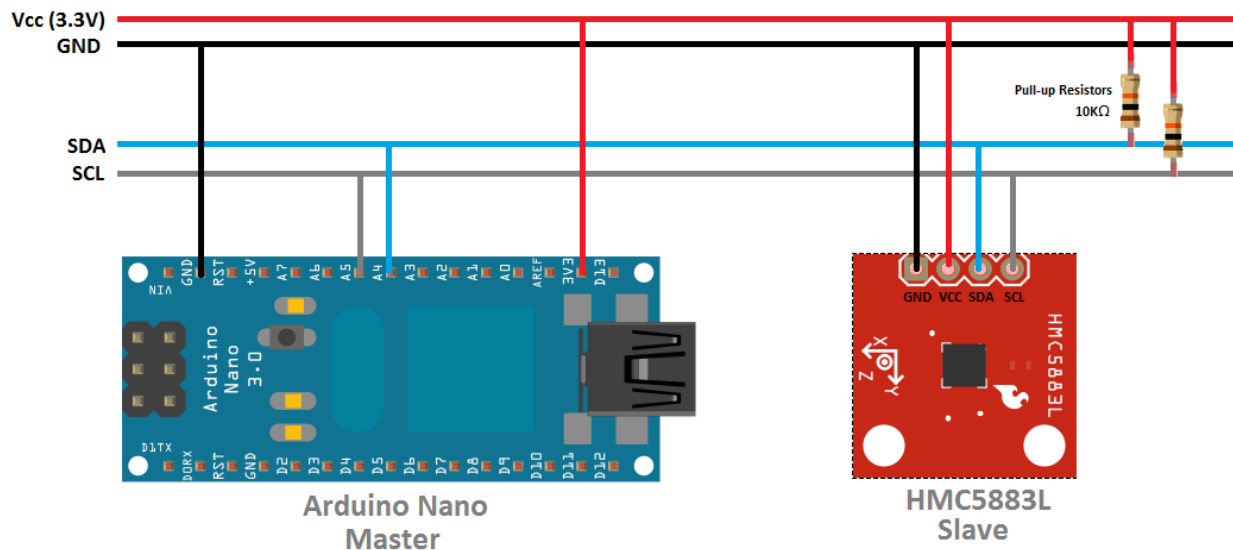


Figure 3.5.2 I<sup>2</sup>C Overview [18]

Slave I<sup>2</sup>C devices come pre-configured with an ID by the manufacturer so that the master device can communicate with more than one slave device by specifying the ID in the I<sup>2</sup>C protocol. This allows the Arduino to communicate with several devices with only two wires.

I<sup>2</sup>C is active low meaning the values need to be put on 0V (LOW) to be considered a "logic 1"; thus pull up resistors are required on both SDA and SCL lines so that its default state is high (3.3V).

Included in the Arduino IDE is the "Wire Library" [18] which enables a simple interface with I<sup>2</sup>C devices.

As the master device the Arduino Nano will do the following to request data from HMC588L compass using the I<sup>2</sup>C protocol:

- 1) Connect to HMC588L (Use ID number)
- 2) Ask for Register X
- 3) End Transmission
- 4) Wait for data response
- 5) Receive Response

The algorithm above combined with other variations of communication is implemented in an open-source library “HMC5883L” which provides a class object to easily request data from the three axis.

To calculate the bearing, extrapolation of the HMC588L data is required. By combining the data of 2 axis of plane, we can calculate the bearing to magnetic north demonstrated by the pseudo-code below:

```
heading = atan2(rawData.YAxis, rawData.XAxis);
```

#### **Applications:**

The small size and relatively low cost of the HMC5883L combined with the simple I<sup>2</sup>C interface makes this device a suitable magnetic sensor for the QuickScan. The HMC5883L is attached onto the QuickScan’s 360° rotational platform enabling mapping with magnetic north information.

## **3.6 Other Components**

There are quite a few other components that have been integrated into our device including the following:

- 12V Li-ion battery
- Pan/tilt rotation cage - built by Dimension Technologies
- Device enclosure and stand
- Other circuitry

Each of the above components is explained in detail in the following sections.



## 4. Mechanical

The QuickScan device consists of four subcomponents; from the bottom up, they are: the tripod, the circuitry black box, the pan-tilt enclosure, and the Kinect. Figure 4.1 shows their placement in the overall device.

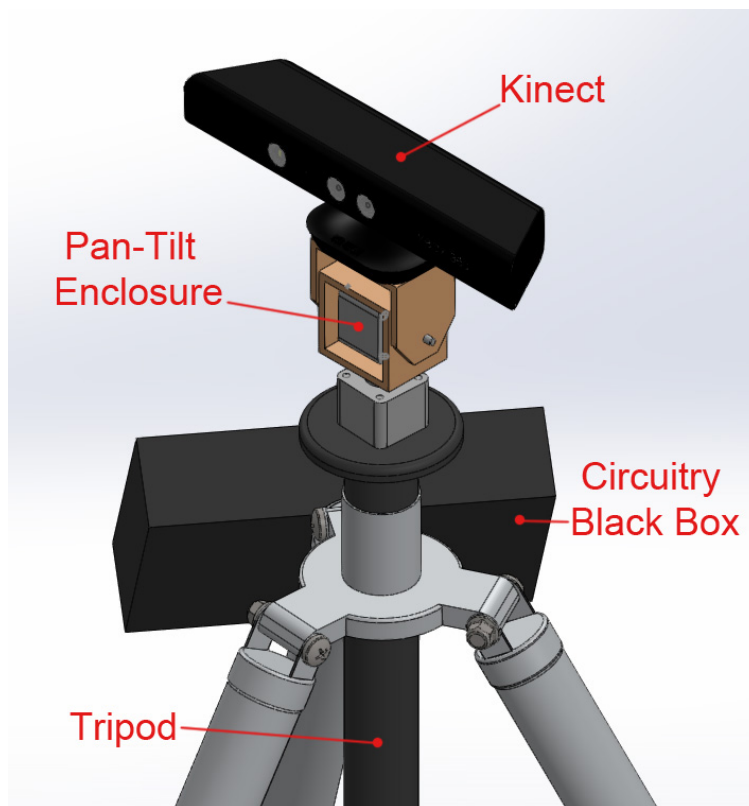


Figure 4.1 Overview of the QuickScan device [28][29]

The tripod used is a standard three-legged tripod, capable of extending and retracting with a height range between 0.58m to 1.6m. The black box contains the Arduino microcontroller, motor driver, external battery, as well as various other circuitry that are necessary for connection. The black box will have a size of approximately 250mm by 170mm, with a thickness of 60mm. The pan-tilt enclosure connects between the Kinect and the tripod; the details and dimensions of the enclosure will be explained below.

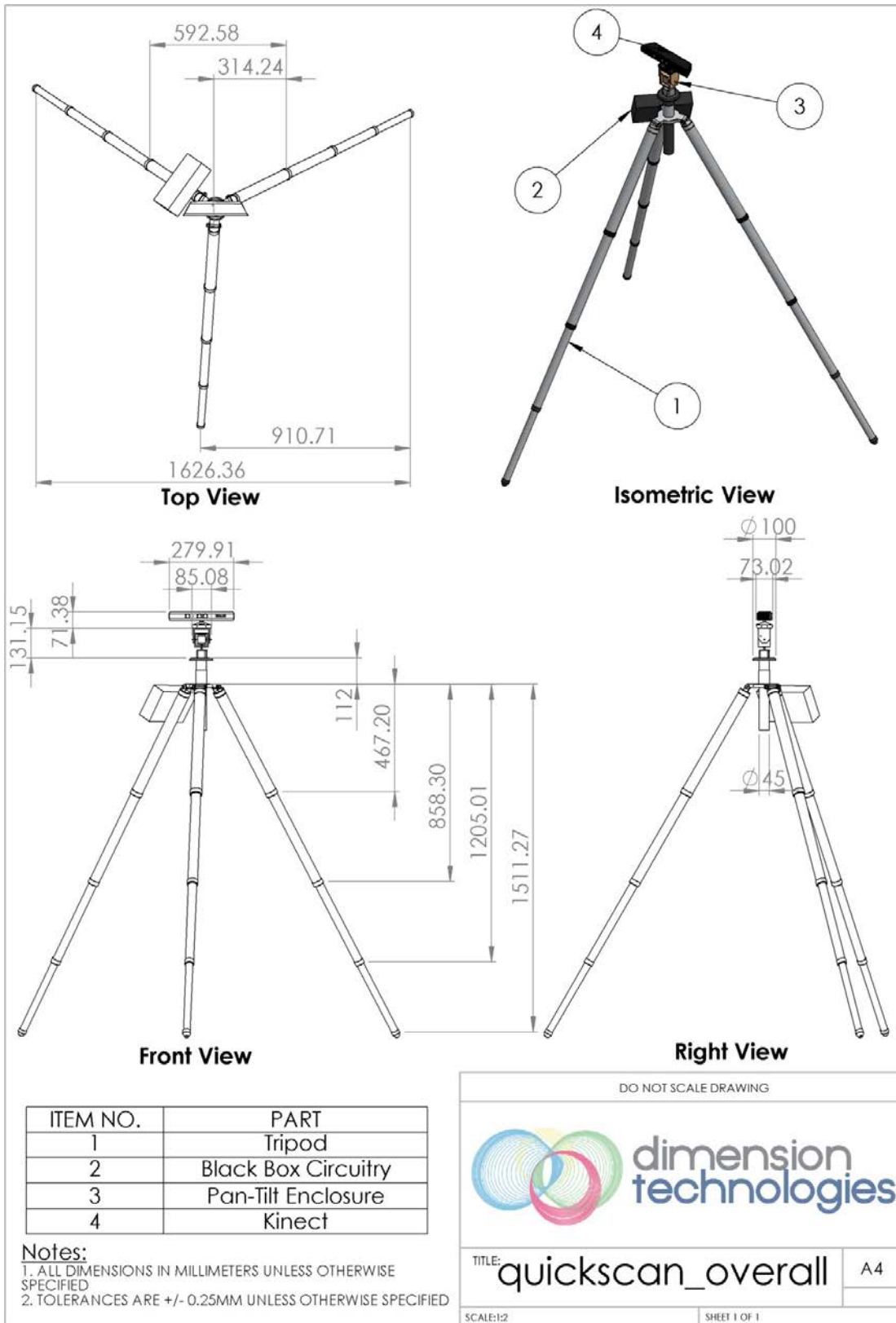
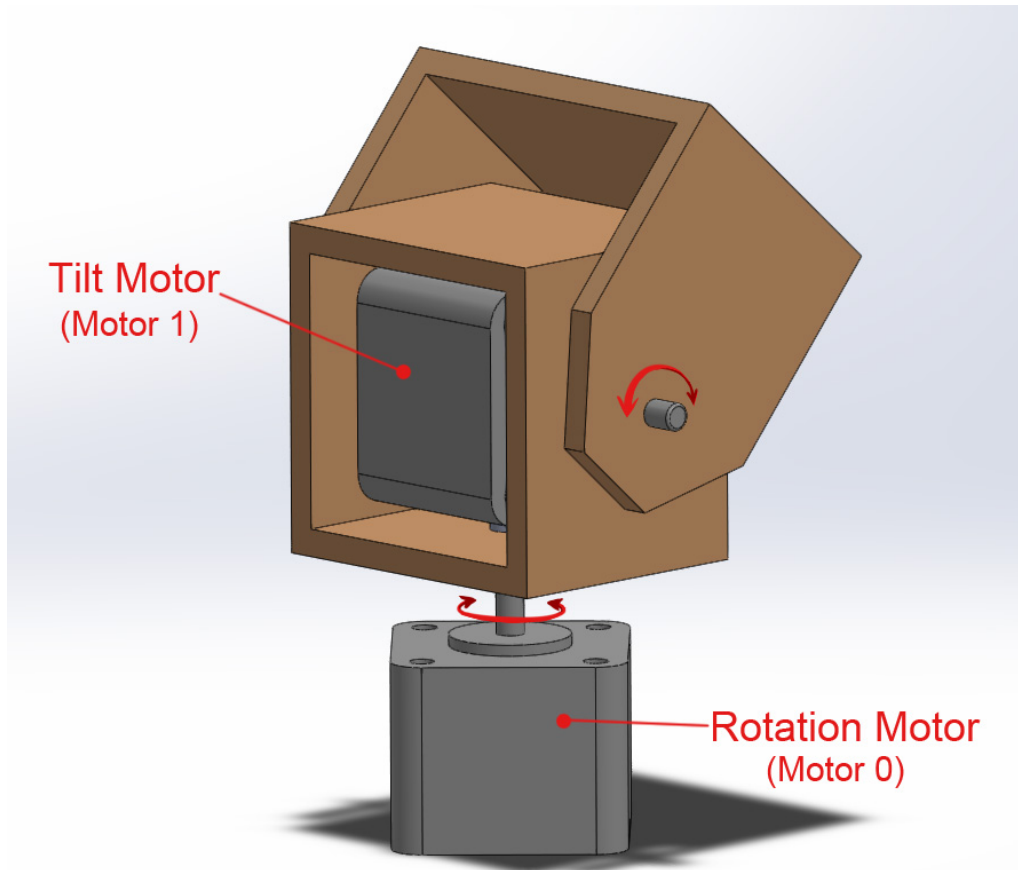


Figure 4.2 QuickScan Diagram [29]



As stated in the hardware section of this document, the stepper motor we are using is the ROB-09238. The body of this motor has a fixed size of approximately 42.3mm by 42.3mm by 34mm. Shown in Figure 4.6, the enclosure has been built to allow comfortable room for the motor to sit in. Please also note that two stepper motors are used as part of the enclosure design: the motor at the bottom will offer horizontal rotation while the motor inside of the enclosure will be able to tilt vertically.



**Figure 4.6 Operation of the two stepper motors**

The pan-tilt enclosure is responsible for connecting the Kinect sensor to the tripod. It will offer 360° rotational freedom as well tilting via the two stepper motors included. The enclosure comprises of two components: the 'house', where the tilt motor can be located, and the 'stand', where the Kinect would sit upon. The figure below shows a mechanical drawing of the pan-tilt structure. The house is approximately 60mm x 60mm x 50mm in size, while the stand is approximately 74mm by 60mm by 50mm (both of which are hollowed out to have a thickness of 5mm). The two enclosure components will be fastened together by a nut and bolt on one side, and by the rod of the stepper motor on the other. The following two mechanical drawings show the dimensions as well as the exploded view of the pan-tilt enclosure and a detailed assembly of the pan-tilt enclosure.

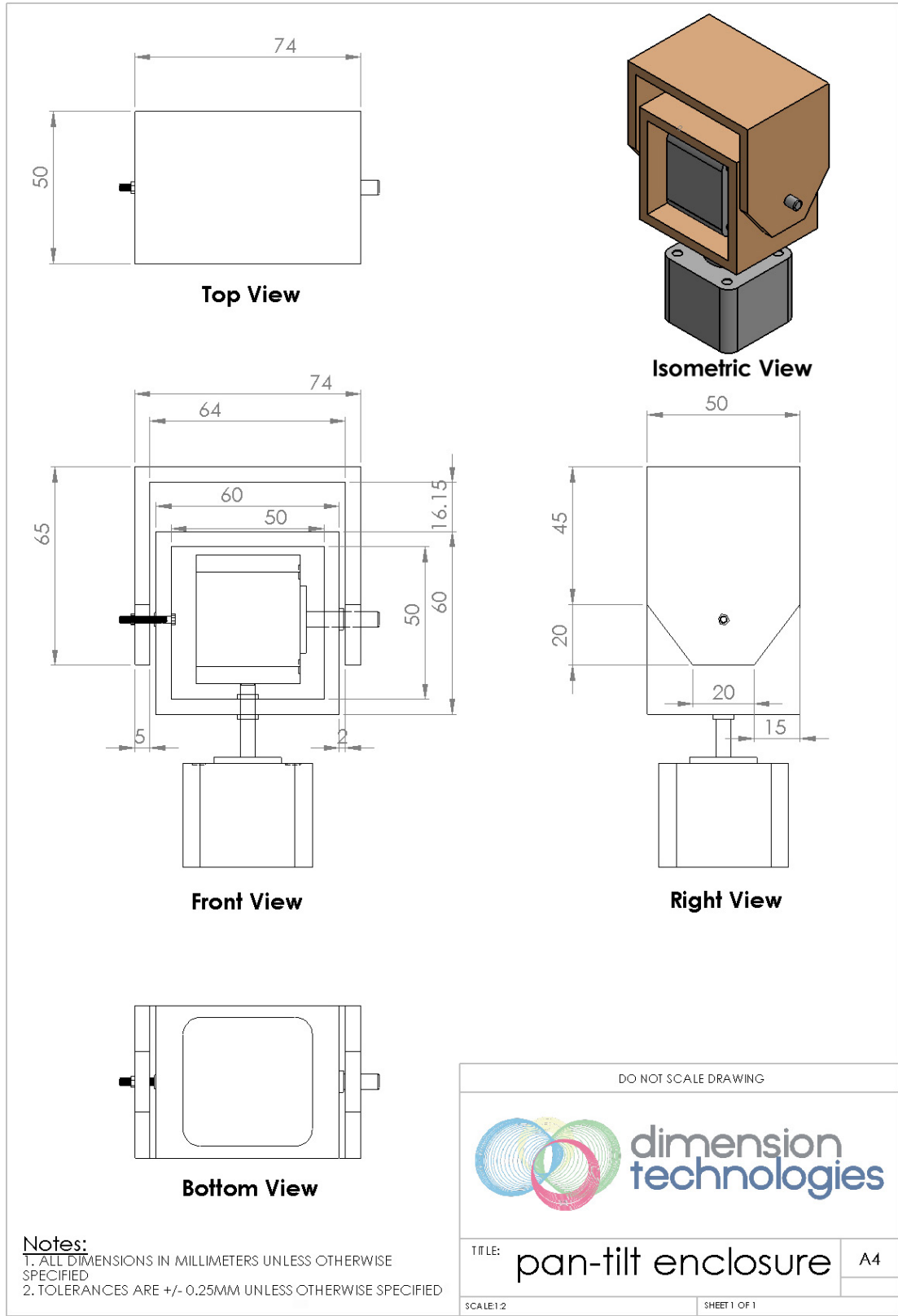
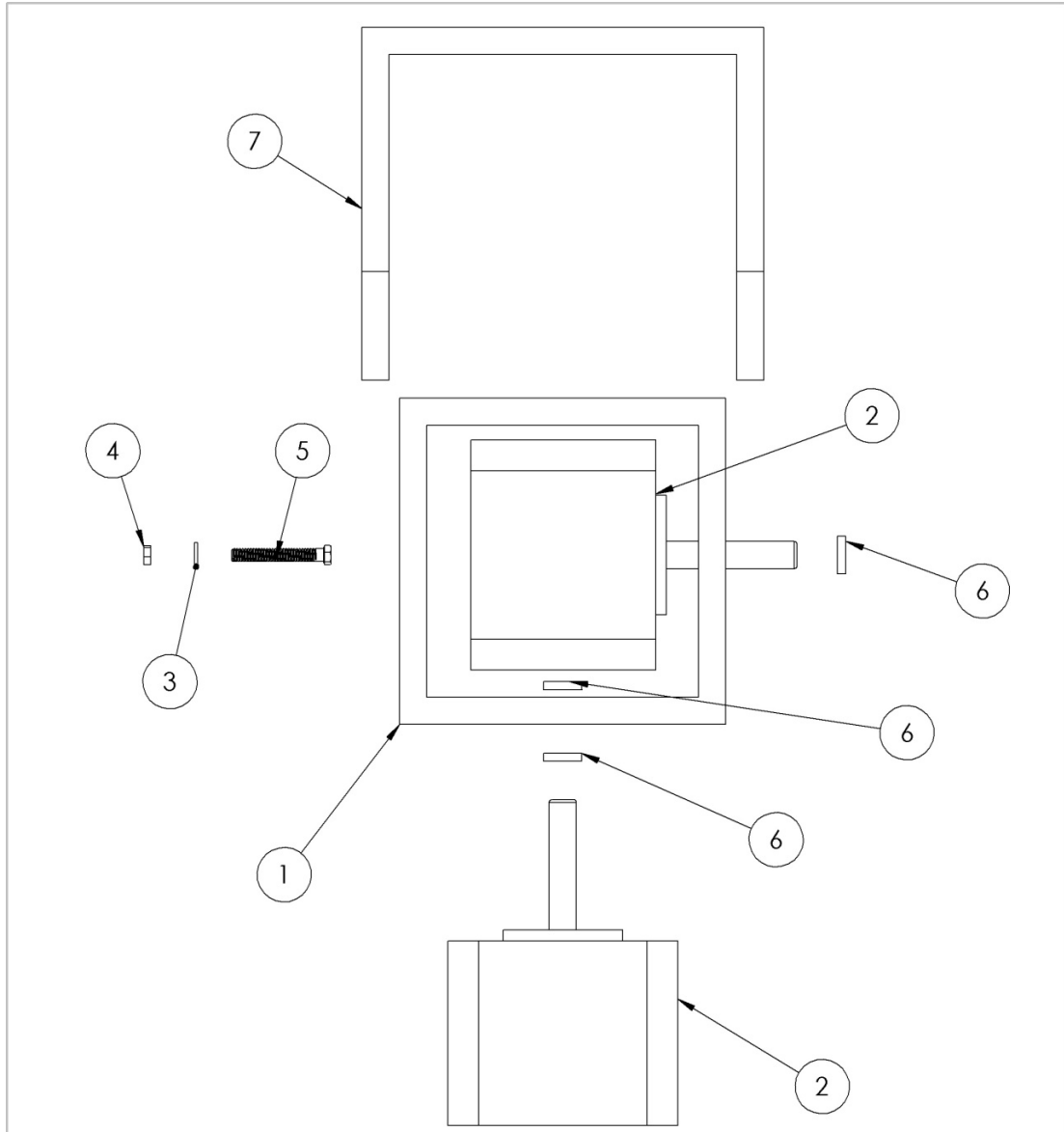


Figure 4.4 Pan-Tilt Enclosure [4]




**Exploded View**

ITEM NO.	PART
1	Enclosure: House
2	Stepper Motor
3	Bearing
4	Nut
5	Bolt
6	Motor Bearing
7	Enclosure: Stand

**Notes:**  
 1. ALL DIMENSIONS IN MILLIMETERS UNLESS OTHERWISE SPECIFIED  
 2. TOLERANCES ARE +/- 0.25MM UNLESS OTHERWISE SPECIFIED

DO NOT SCALE DRAWING



dimension technologies

TITLE: pan-tilt enclosure exploded view	A4
SCALE:1:2	SHEET 1 OF 1

**Figure 4.5 Pan-Tilt Assembly Diagram [4]**

The next figure shows a mechanical drawing and dimensions of the ROB-09238 stepper motor.

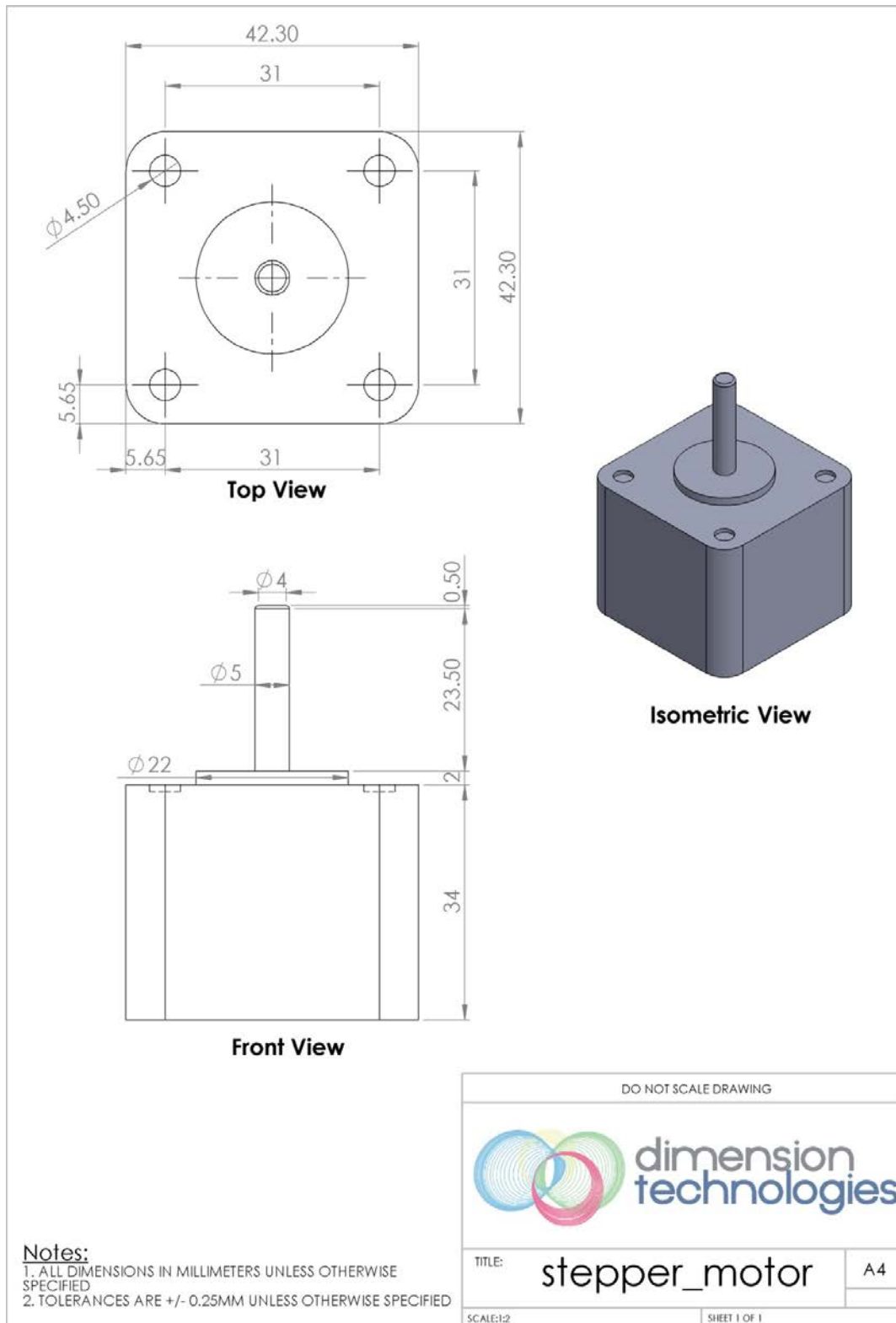


Figure 4.6 Motor Mechanical Diagram [4]

## 5. Software Processing

### 5.1 Data Processing

The key feature of the QuickScan is to automatically capture a full 360° of a room and since the IR sensors on the Kinect are only able to capture the forward facing data, a full view of a room, the Kinect camera has to be rotated horizontally and vertically. As a result, mathematical compensation is required to adjust the virtual camera location in our software to ensure consistency between capture points when the Kinect camera is moved.

To solve this problem, forward kinematics will be used to determine the new sensor coordinates in frame  $\{3\}$  in reference to frame  $\{0\}$  as shown in the diagram below:

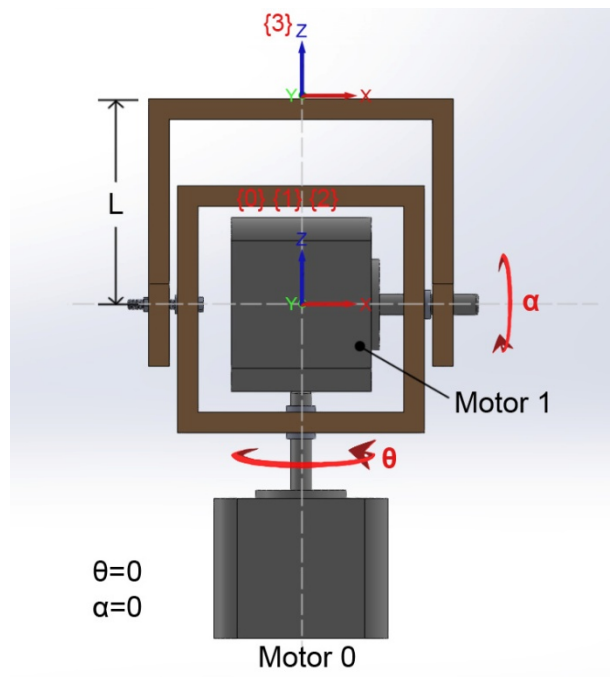


Figure 5.1.1 Forward Kinematics

Frame {0} is the stationary frame of reference located in the cross section between the center line from the rotation axis of Motor 0 and the centre line from the rotation of axis of Motor 1. When frame {1} is located at the same location when  $\theta = 0$ , and rotates on the z-axis with angle  $\theta$  in reference to frame {0}; similarly, frame {2} rotates on its x-axis with angle  $\alpha$  in reference to frame {1}. Finally frame {3} is shifted up distance "L" on the z-axis in reference to frame {2}.

The Kinect sensor's location in relation to the final platform, frame {3} located at point  ${}^3P = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}$ ,

where  $P_x$ ,  $P_y$  and  $P_z$  are minor offset adjustments that are used fine tune and calibrate its data capture.

The new camera coordinates  ${}^3P$ , affected by the vertical and horizontal rotation  $\alpha$  and  $\theta$  are given in the equations below:

$${}^3P_x = P_x \cos(\theta) - P_y \cos(\alpha) \sin(\theta) + L \sin(\alpha) \sin(\theta) + P_z \sin(\alpha) \sin(\theta)$$

$${}^3P_y = P_x \sin(\theta) + P_y \cos(\alpha) \cos(\theta) - L \sin(\alpha) \cos(\theta) - P_z \sin(\alpha) \cos(\theta)$$

$${}^3P_z = L \cos(\alpha) + P_z \cos(\alpha) + P_y \sin(\alpha)$$

One problem that can arise from changing the axis happens when the images are brought together, see Figure below.



Figure 5.1.2 Sample stitch of two snapshots

The present view shows the joint part when two shots are brought together. It is quite obvious that there are problems such as there are two poles when they should be put together in an overlap. This happens due to the fact that the Kinect is perfectly in the centre of the platform when it turns.

To resolve this issue, we shall first attempt to find the centre of the Kinect camera as described in the test plan and then through trial and error we will attempt to place the Kinect at the centre of the platform.

## 5.2 Software Processing

### General Requirement

- Processing 1.5.1 this is a programming language for visual display used to process the data retrieved from the Kinect [10]
- OpenNI 1.5.4.0, this provides tools used to capture the point cloud data for the Kinect [9]
- NITE 1.5.2.21, PrimeSense's middleware which allows computers to perceive the world in 3D
- Sensor 5.1.2.1, PrimeSense's driver for the Kinect
- SimpleOpenNI, this is an OpenNI and NITE wrapper for Processing
- KinectOrbit, a camera control library for processing
- ControlP5, this is a Graphic User Interface Library for Processing [11]

### Processing Software System Overview

Processing is a open source software that is designed for visual design. The software itself is built on Java although it has a simpler programming syntax and graphics programming model. It includes a feature called the sketchbook which is an alternate to an IDE for organizing projects.

We are using processing to control the Kinect and to access the data that the Kinect is capturing. In order to do this we incorporate the SimpleOpenNI library to assist us. As mentioned above, SimpleOpenNI is an OpenNI and NITE wrapper for processing.

Processing is used to write the software that processes the point cloud data captured and then results into a \*.ply file to be viewed in MeshLab.

### OpenNI and NITE

Open Natural Interaction (OpenNI) is an industry led, nonprofit organization that designs frameworks that allow devices such as the Kinect to recognize natural interactions. One of OpenNI's main members is PrimseSense which is the company that designed the technology



used for the Kinect for motion sensing. Recently they released an open source driver that tracks these motions called NITE. With these technologies, OpenNI provides features such as hand gesture recognition or hand control which allows you to control digital devices with your bare hands. Also they provide features such as the full body control, which allows you to use your entire body in a full body video game.

With the SimpleOpenNI library, we create a variable called *kinect* and then with it we enable its depth sensor and its RGB sensor. The depth sensor, through the IR projectors, projects a pattern of infrared light beams into the environment in front of the sensor. Therefore with every pixel it captures, it is able to detect how far away that pixel is from the camera thus giving us the z coordinate. Then going through each pixel that is being captured, the coordinate of each pixel as well as its RGB value will be captured and will be stored separated into two vectors.

## Meshlab

Meshlab is an advanced 3D mesh processing software. Meshlab takes the point cloud data with all its coordinates and projects a 3D view of the coordinates. This is the software that we will use to view our final results. A sample of which is seen in the Figure below. We will also use this to do some post analysis for surface recreation and filtering.

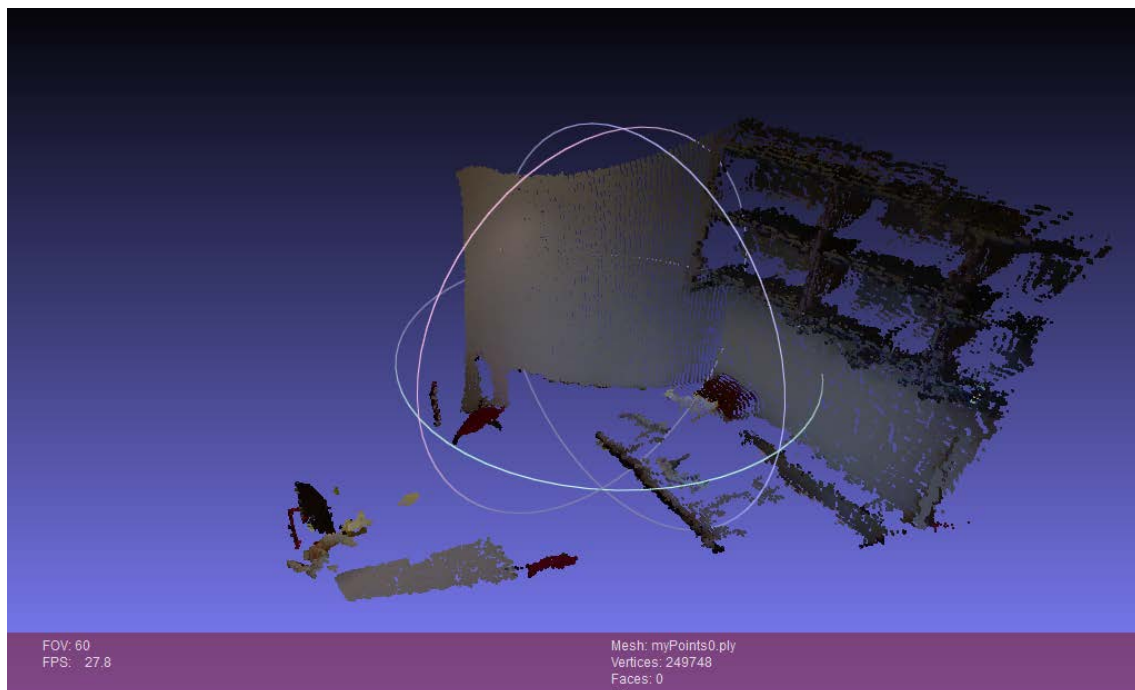


Figure 5.2.1 MeshLab Output



### 5.3 QuickScan Procedure

The procedure for capturing the images is portrayed in the Figure below.

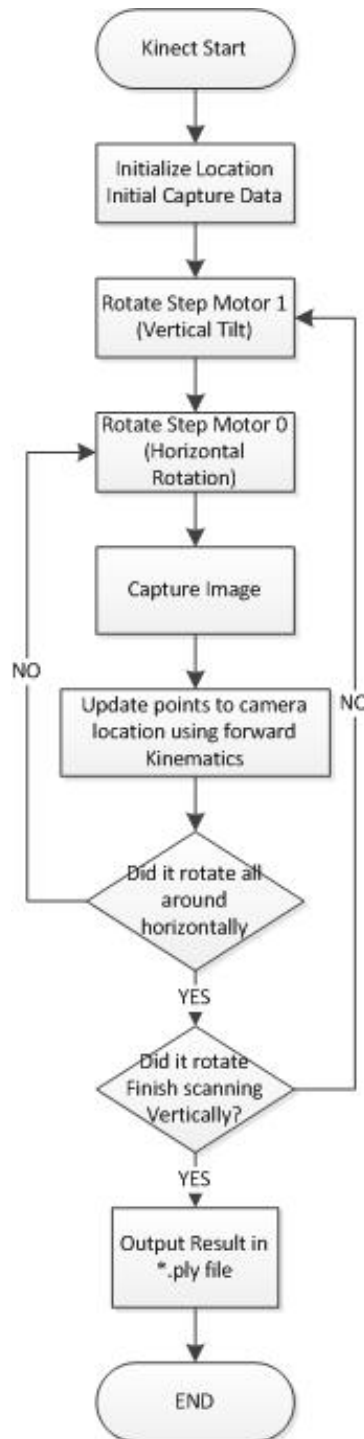


Figure 5.3.1 QuickScan Procedure

Since the Kinect is working with the stepper motor at certain stages of the procedure it is waiting on the stepper motor to finish its task.

1. The first step is to initialize the Kinect. Once that is ready we check from the compass where north is and then rotate the camera via the stepper motor to face north.
2. The next step is to rotate motor1 (vertical) to its proper vertical location (upper view, straight view, and the lower view).
3. Then motor0 (horizontal) is to be rotated to the designated degree.
4. The image is then captured with its RGB value and its coordinates.
5. The axis of the output vector is rotated to the camera view through forward kinematics and then the RGB values and the coordinates are stored into the output vectors.
6. Next check if it has captured every view all around horizontally, if yes proceed. If not, go to step 3.
7. Check if it has captured all the vertical views. If yes proceed to the last step, if not go to step 2.
8. All the data in the vectors are exported onto a \*.ply in a particular format seen below.

Figure 5.3.2 depicts an example to show the procedure to which a room is being scanned.

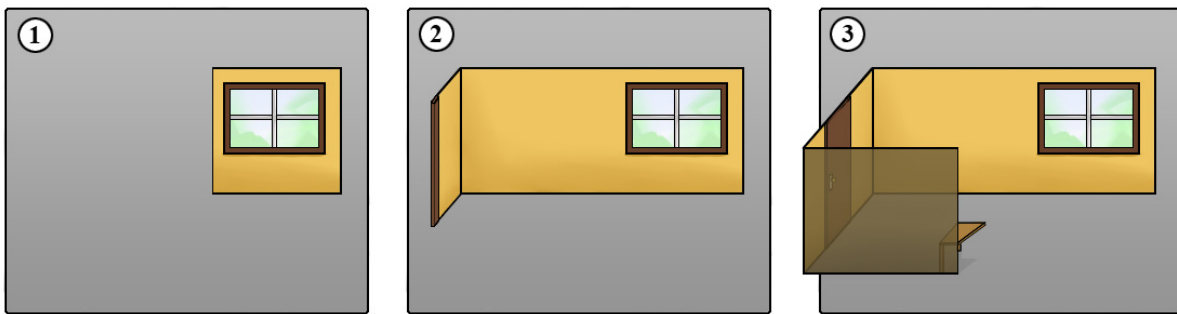
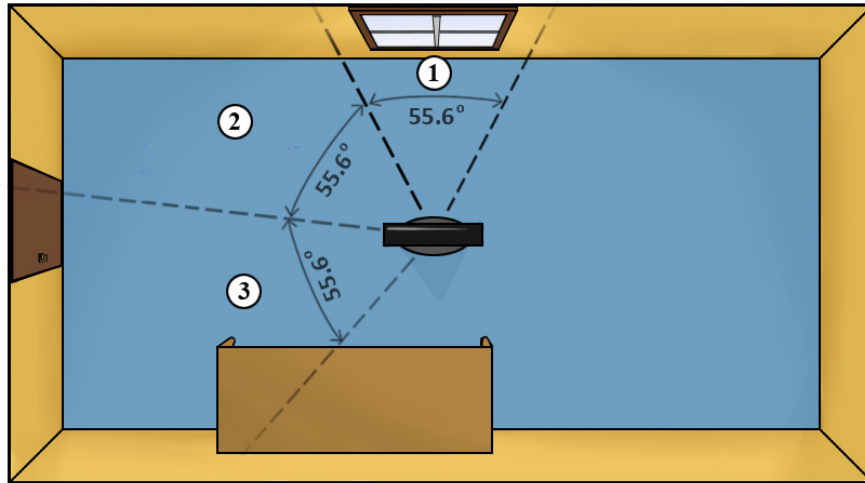


Figure 5.3.2 Illustration of QuickScan Process

The above image graphically illustrates the QuickScan mapping process. The three images on the bottom depict the progress as the room is being scanned. The images that are taken are being joined together via the data processing theory that was described above by adjust the axis

## 5.4 Graphical User Interface (GUI)

Our project's Graphical User Interface (GUI) has been developed by using the ControlP5 [11] library; one of the GUI libraries available for Processing. ControlP5 is a controller library for Processing that can be used in authoring, application, and applet mode. Controllers such as sliders, buttons, toggles, knobs, text-fields, radio-buttons, and check-boxes amongst others are easily added to a processing sketch. They can be arranged in separate control windows and groups. ControlP5 offers a range of controllers that allow developer to easily change and adjust values while the developer's sketch (refer to processing section) is running.

In our project, we have made use of the ConralP5 library to create the QuickScan GUI. The GUI will be the main venue of interaction between the user and the QuickScan device. The GUI supports the following features:

- COM port selection to QuickScan device
- Live image of QuickScan process
- 'Start' and 'Stop' buttons
- Image panning and rotation features

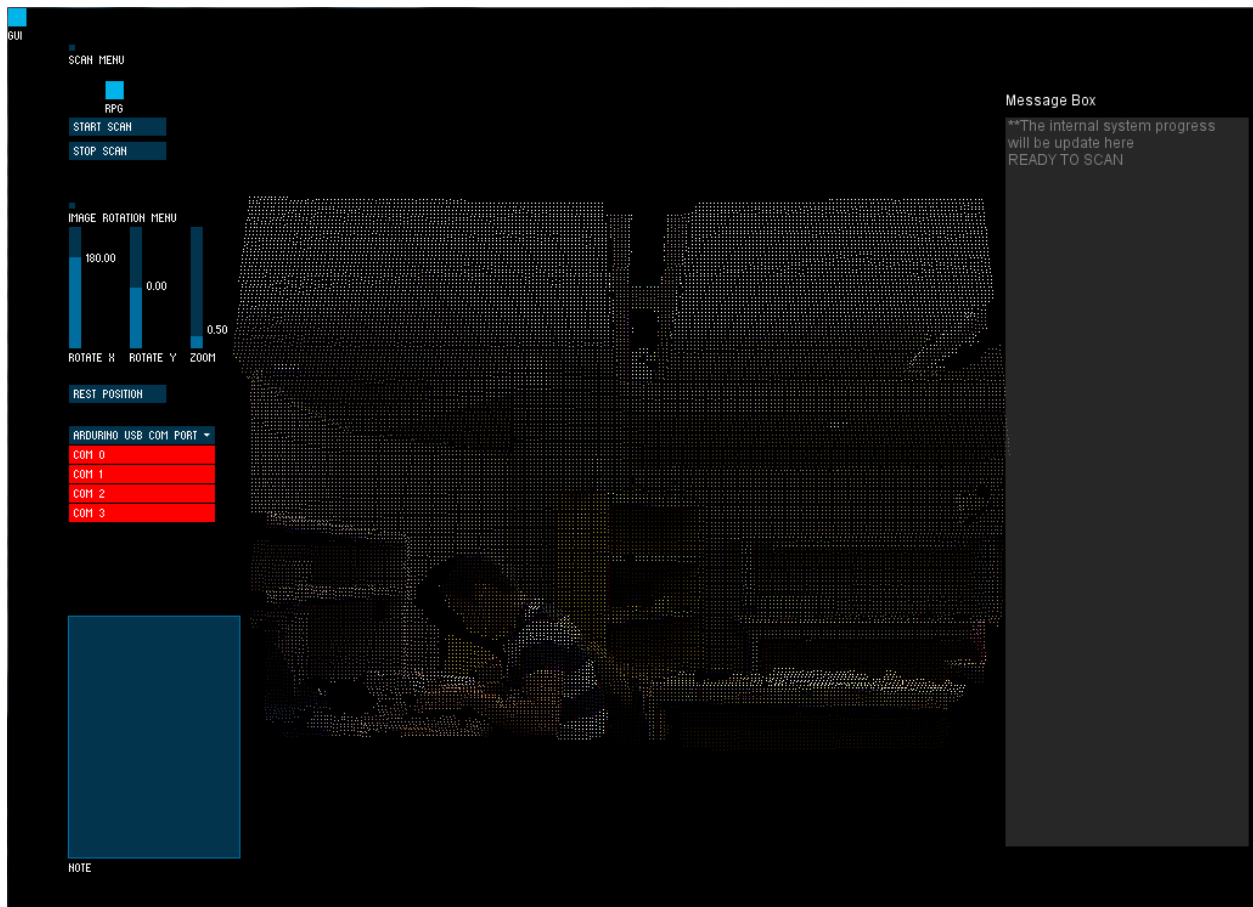


Figure 5.4.1 QuickScan GUI

## 6. Arduino Software

### 6.1 Serial Communication Protocol

#### **Arduino Communication:**

The ATmega328 microcontroller onboard the Arduino Nano provides UART TTL (5V) serial communication. The serial data from the Atmega328 is relayed to the FTDI USB chip to channel data over USB by using the included FTDI drivers (included in the Arduino software) to provide a virtual com port onto the computer. Processing can then access this virtual com port to communicate with the Arduino.

#### **Overview:**

Before the Arduino and Processing are ready to communicate with each other, they must synchronize and indicate that they are ready to receive data. This is important since there is a one to two second delay when Processing is starting and any data packets out of order will affect the communication protocol between the two modules.

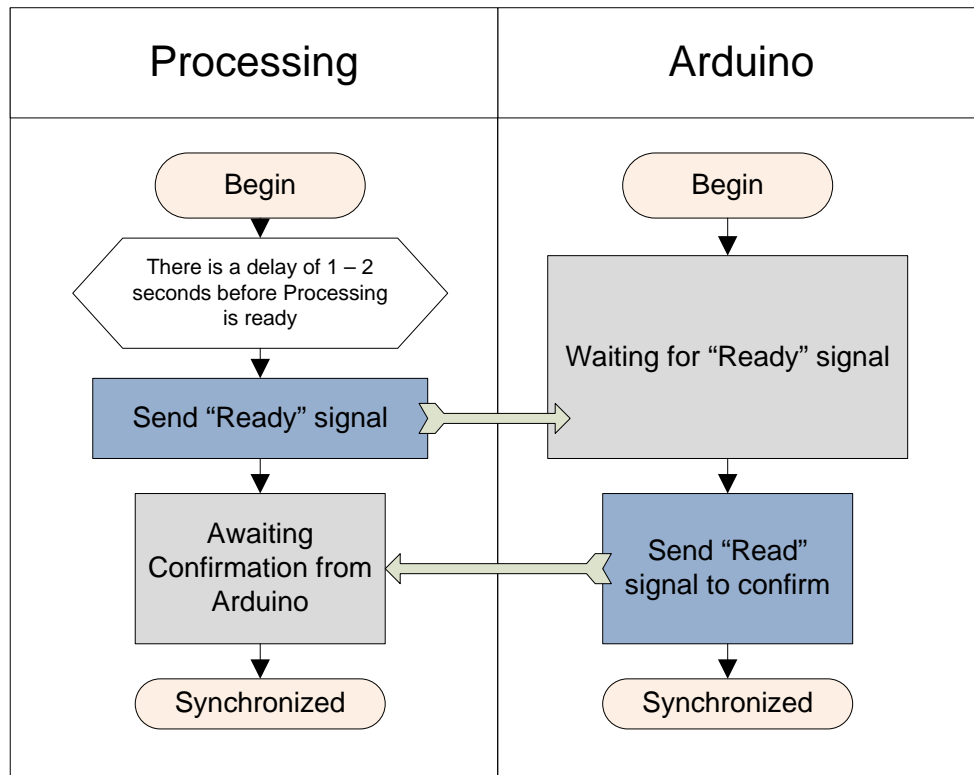


Figure 6.1.1 Arduino & Processing Synchronization

Once the synchronization is established between Processing and the Arduino, Processing can request data by sending a specified “request byte” and the Arduino will perform the desired and confirm that the task has been completed by returning a “confirmation” byte.

Below is the protocol required for turning stepper motor 0 into a specific angular position:

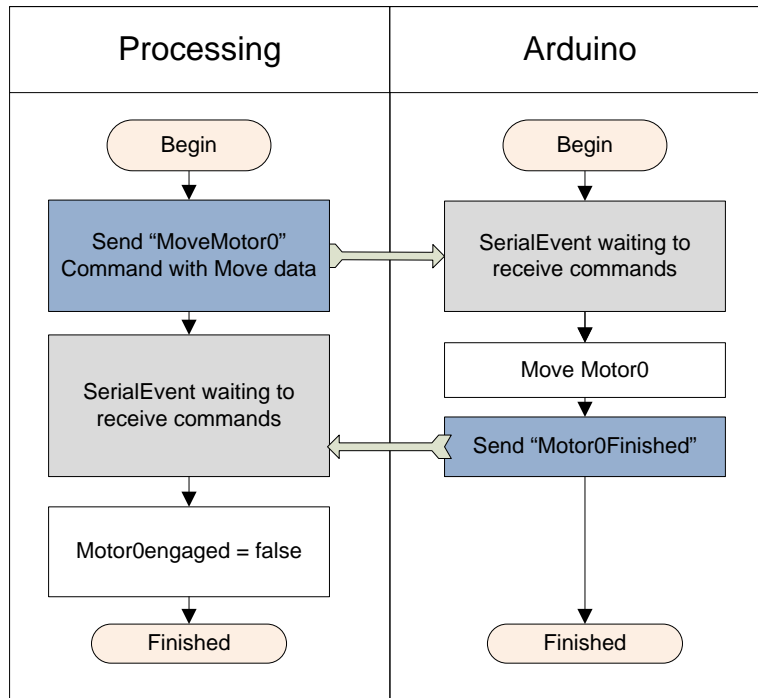


Figure 6.1.2 Stepper Motor 0 Protocol

Figure 6.1.3 shows the protocol required for requesting the magnetic bearing data from the Arduino:

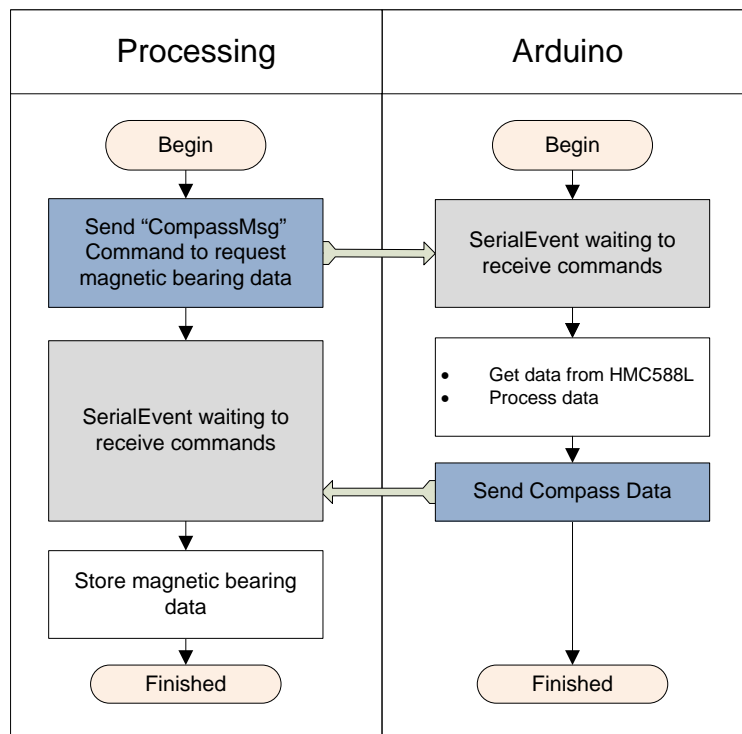


Figure 6.1.3 Magnetic Bearing Protocol

Figure 6.1.4 is the protocol required to turn the stepper motors ON or OFF. This is an important feature since the QuickScan's stepper motors draw ~300mA of current each; this is required to keep its position and maintaining a strong holding torque. However, once the stepper motors are completed with their tasks, the motors will be turned off to preserve battery life.

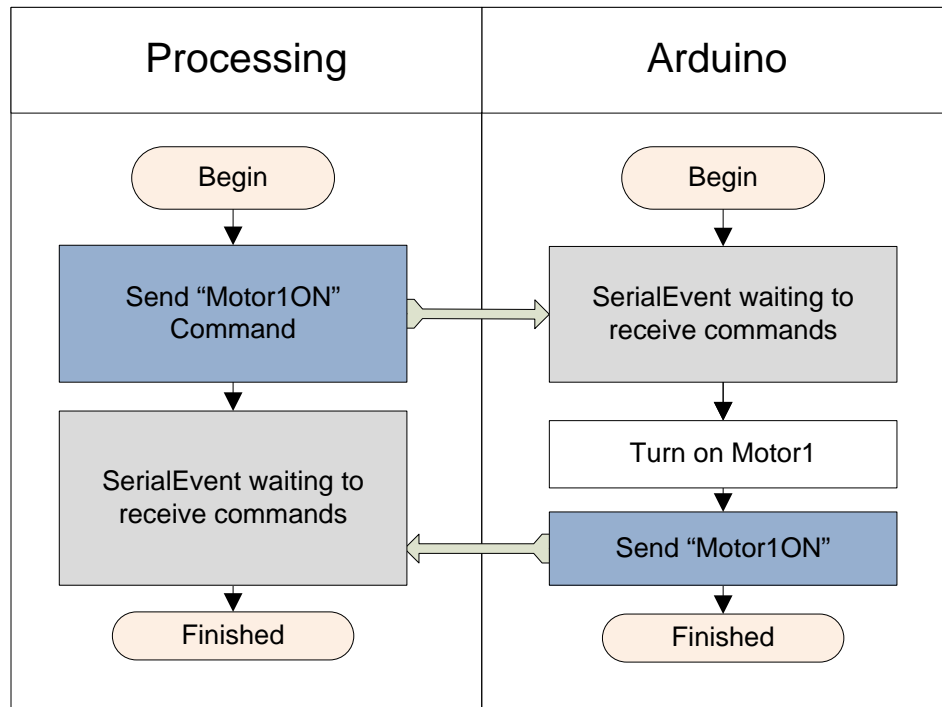


Figure 6.1.4 Stepper Motor 1 Protocol



## 6.2 Arduino Microcontroller Logic

### Overview:

To begin, the Arduino initializes all its components: Serial communication, HMC5883L compass, and the two EasyDriver for the stepper motors. Then it will synchronize with Processing by waiting for a “Ready” message before entering the main loop.

As explained in the communication protocol between the Arduino and Processing, after the Arduino has received the “Ready” message, it will enter the main loop where the its primary function is to run the two stepper motors.

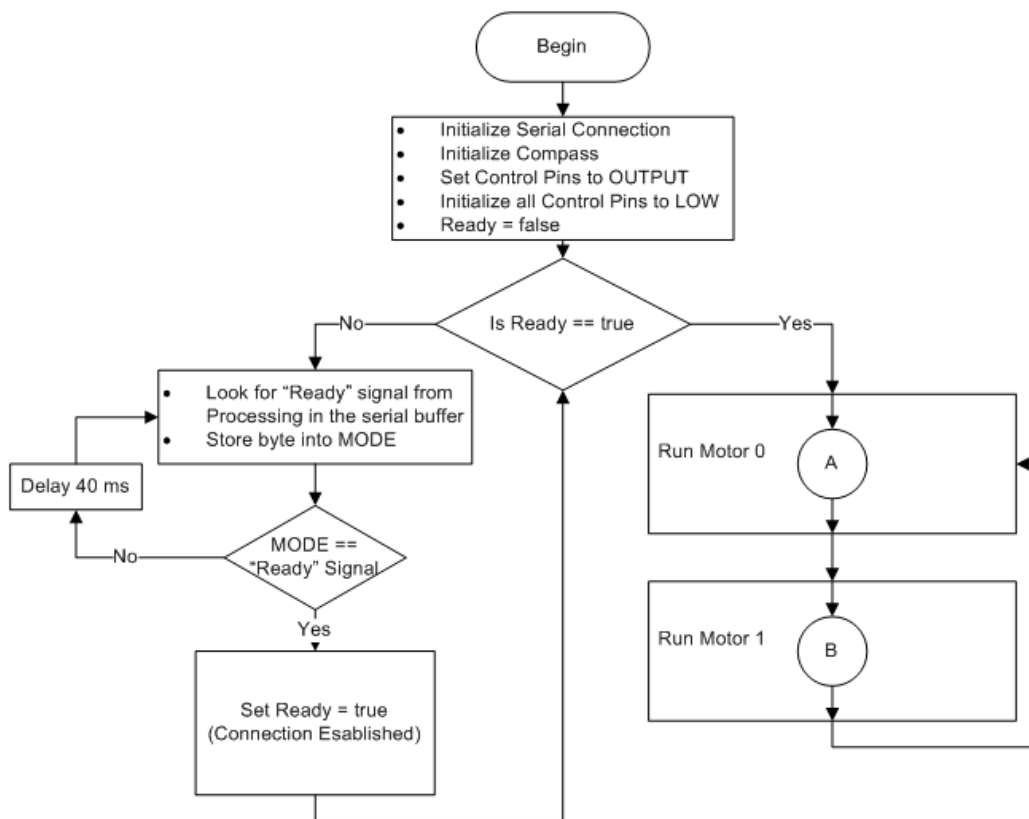


Figure 6.2.1 Arduino Logic

Details of the logic flow in controlling Motor0 and Motor1, (A) and (B) are explained in Figure 6.2.2:

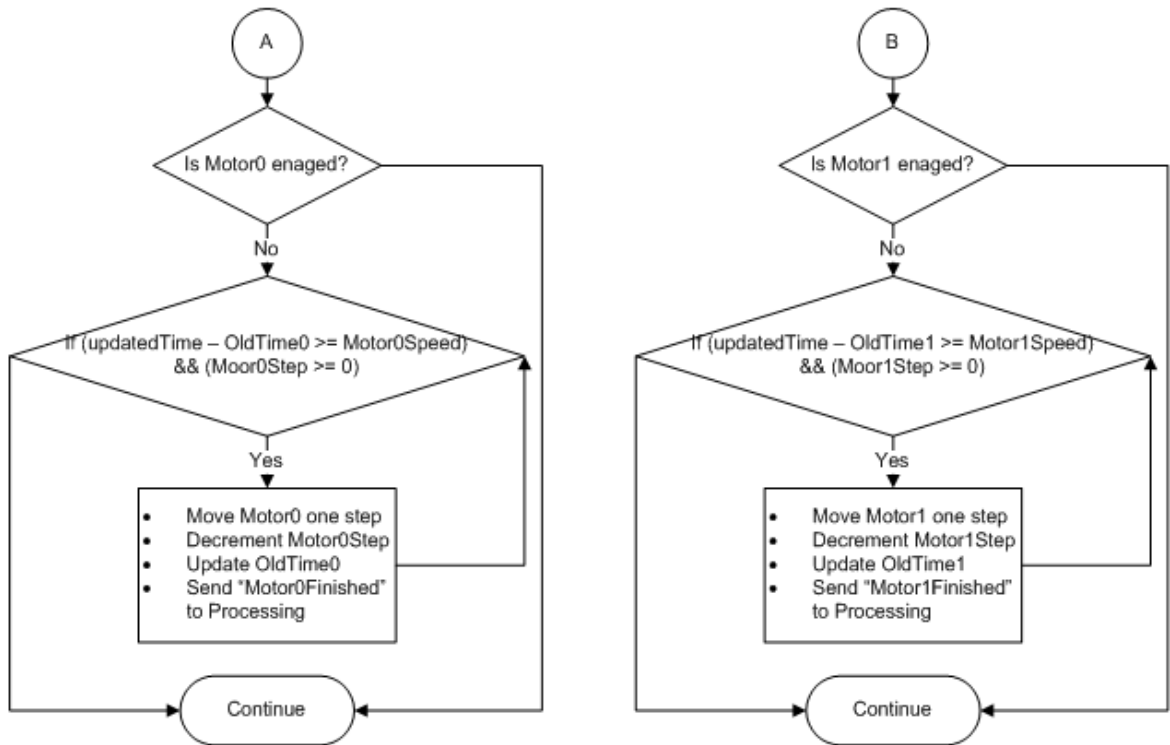


Figure 6.2.2 Motor Control Logic

To implement asynchronous motor control, the *delay()* function cannot be used to create the pulses for the step inputs to send to the EasyDriver. Instead, by using the *micros()* function to store and update the time (resolution of 4 microseconds), the time difference can be calculated and checked against the *Motor0Speed* variable to determine if it is time to send another pulse to the EasyDriver moving the stepper motor one step while the *Motor0Speed* adjust the frequency pulses to the step input. The number of steps and is updated through the *serialEvent()* interrupt and decremented accordingly to ensure the motor moves to the specified angle.

In addition to the main loop driving the two stepper motors, logic below (*serialEvent()*) runs inside an interrupt routine and is called every time information is received from the serial connection. This subroutine ensures that task requests from Processing are processed according to the request message as explained previously in the communication between Processing and the Arduino.

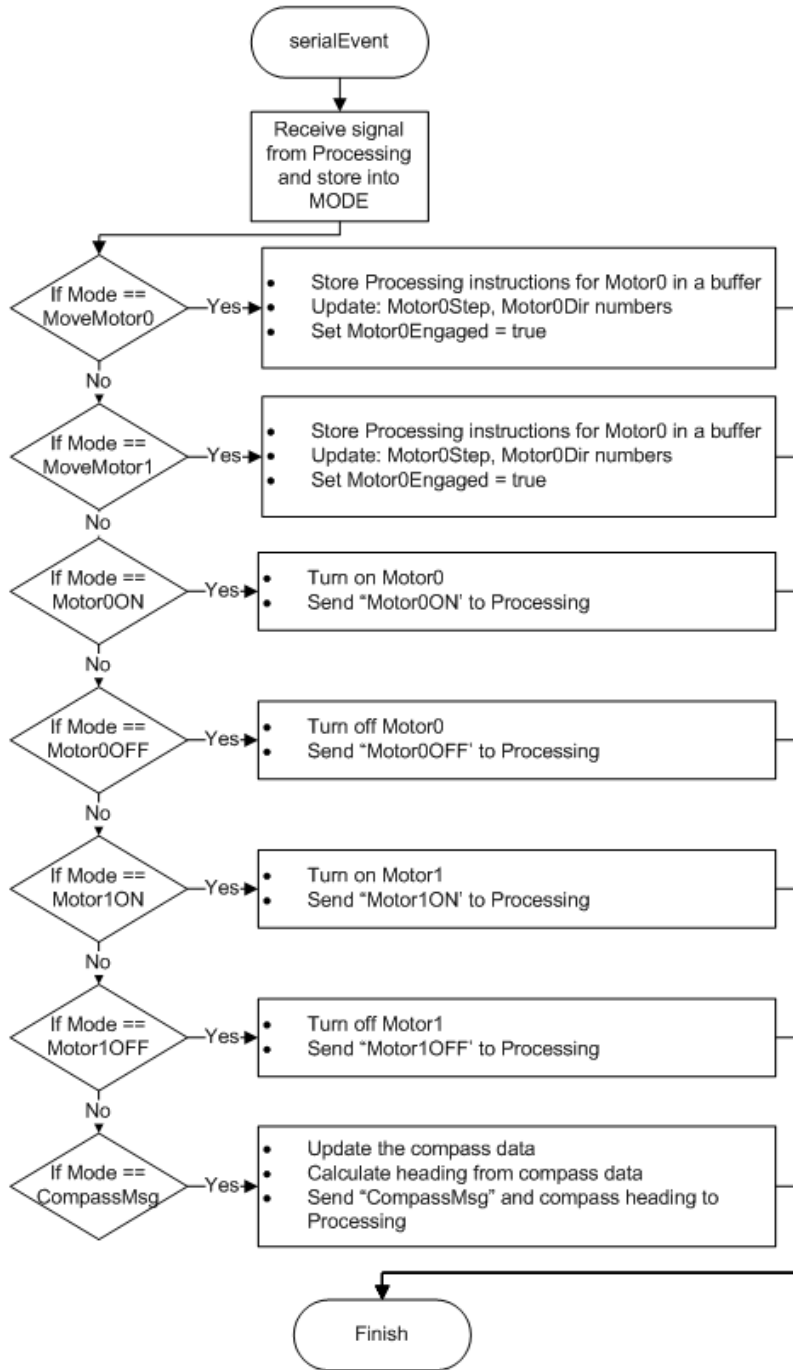


Figure 6.2.3 Motor Interrupt Subroutine Logic

# 7. Electrical

## 7.1 Circuit Design & Topology

The figure below shows all the hardware components connected. It can be observed that the Arduino Nano and the HMC5883L compass receives its power from the USB connection and the 2 stepper motor drivers and the stepper motors receive 12V from an external battery.

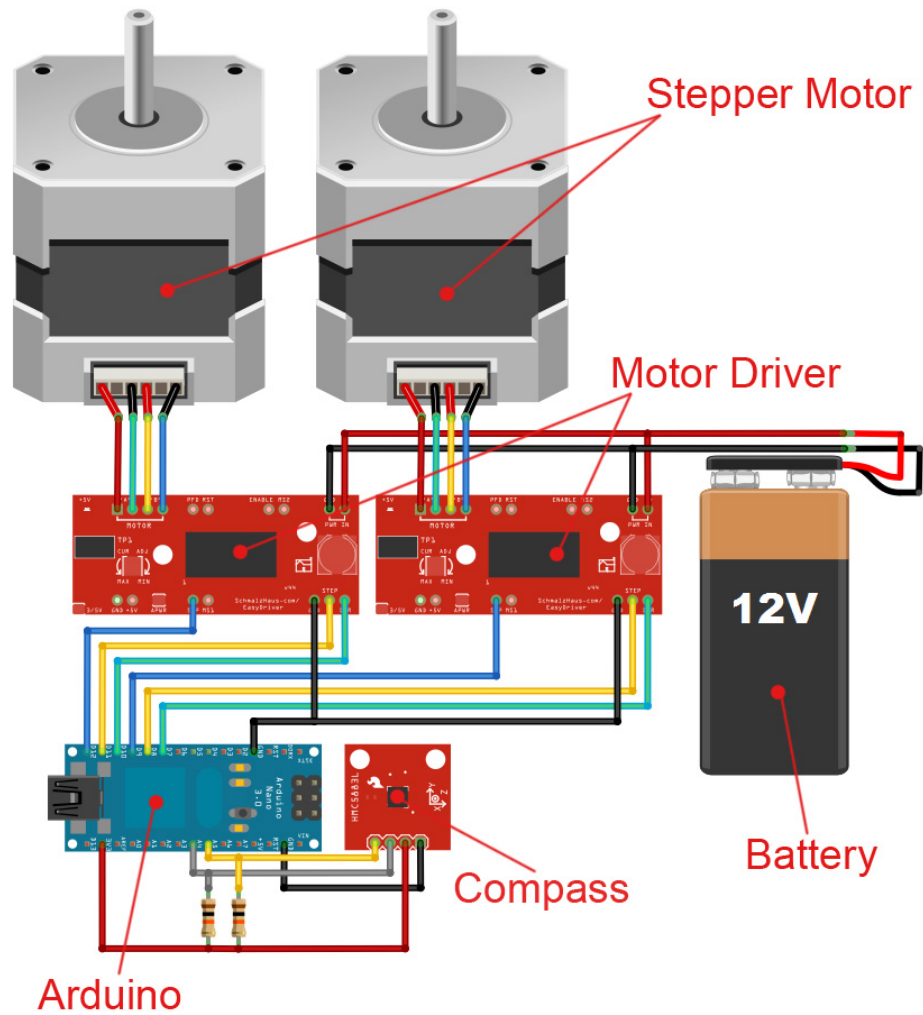


Figure 7.1.1 Circuit Schematic

We plan to integrate each of the hardware components seen above into a PCB module design. This will minimize the size of our electronic circuitry and aid in the placement of our circuitry module on the physical device. The connections between each of the device pins are shown in the figure below. *(Note: Wiring colors in the figure below are consistent with the above figure. i.e. Red indicates power)*

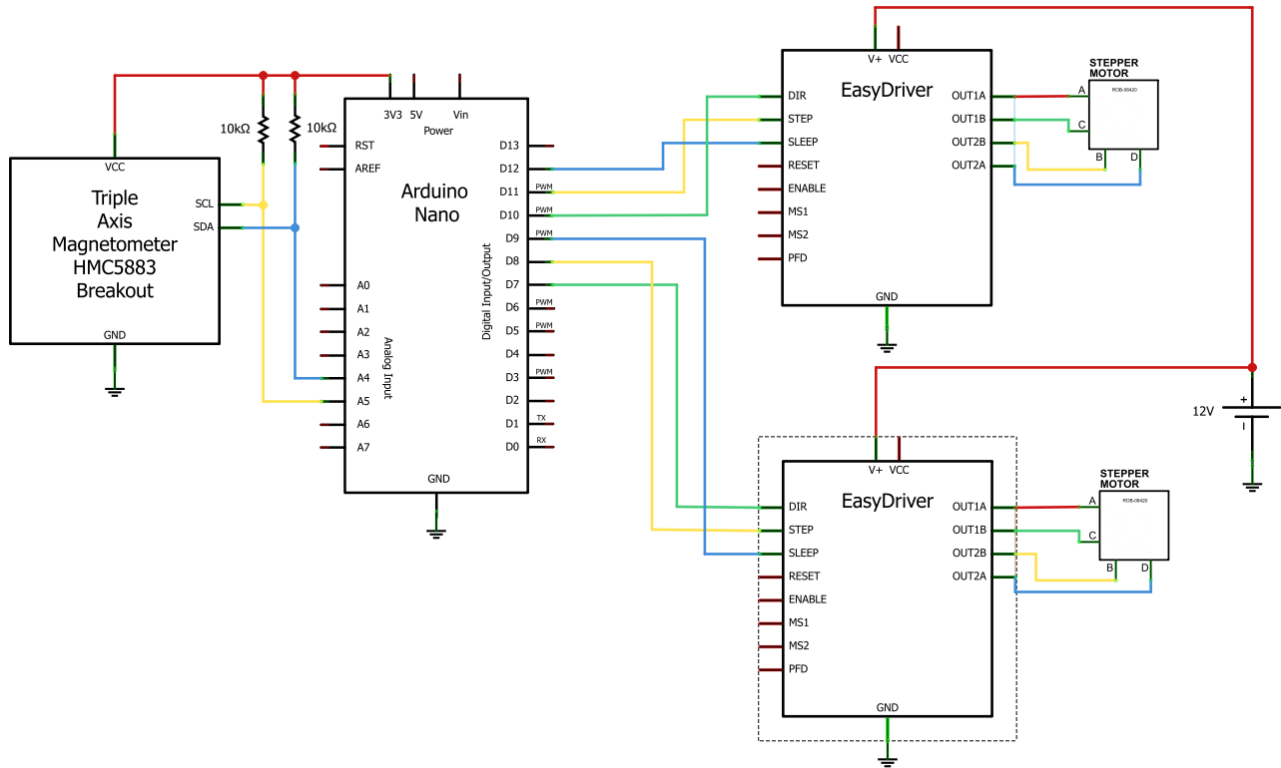


Figure 7.1.2 Wiring Diagram

## 7.2 Battery Powering Considerations

As mentioned in the system overview, the QuickScan module will be battery powered. The following table identifies the operating voltage and current for each of the subcomponents in our design. Note that the current values reflect the case when the device is fully operational and not in its idle mode. Also, the Arduino has not included in the following table as it will be powered through its USB connection and does not require power from the battery. The Arduino has an on-board regulator that will power the compass module which is also not included below.

Table 7.2.1 Component Voltage & Current

Device	Voltage (V)	Current (mA)
Kinect	12V	500
Stepper Motor 1	12V	333
Stepper Motor 2	12V	333
Motor Driver 1	12V	0.2
Motor Driver 2	12V	0.2

From the above table we find that the maximum total current is **1166.4 mA**. Given the fairly high value of current, we decide to use a 12V lithium-ion battery. This battery is rechargeable and shown in the figure below.



Figure 7.2.1 QuickScan Battery

The battery has the following features:

- 12V output
- 3.8Ah
- Rechargeable
- Li-ion

The total estimated power dissipated by the battery is:

$$P = VI = (12V)(1.1664A) = 13.99W$$

A fairly accurate estimate of the operating duration of the QuickScan module with this battery is:

$$T = \frac{3.8Ah}{1.1664A} = 3.25 \text{ hours} = 3 \text{ hours, } 15 \text{ minutes}$$

It is quite clear that this battery is a good option and will enable the user to operate the device for a fairly long time. In practice, the device would operate longer than the time calculated above. This is because the above numbers have been calculated using maximum current values which indicate current values in operation modes where all the devices are fully functional. In reality, there are times where certain device will be in an idle or sleep mode, thus greatly reducing the current draw.

## 8. Test Plan

### Test Plans – Arduino Serial Communication Protocol – Unit Testing

Task: Test the communication protocol and hardware functionality

Task Steps:

1. Using processing, send a message to activate motor 0 and observe if the desired response is initiated.
2. Using processing, send a message to activate motor 1 and observe if the desired response is initiated.
3. Using processing, send a message to activate motor 0 and motor 1 simultaneously and observe if both motors are behaving correctly and satisfies parts 1 and 2.
4. Using processing turn ON and OFF motors 0 and 1 and observe if the desired response is initiated.
5. Using processing, request for the north bearing data from the compass and observe if the correct data is received.

### Test Plans - Processing - Unit testing for the Kinect (without rotation)

Task: Test the depth sensor

Test Steps:

1. Place objects in front of the Kinect at different distance from it
2. Run the program
3. Check if the objects are at the right proportion

Task: Test the distance measuring feature

Test Steps:

1. Using a measuring tape to find a point that is 10 metres directly in front of the camera
2. Place an object at that location
3. Check to see if the output value is 10 metres

Task: Test for the centre of the Kinect

Test Steps:

1. Place a flat object at a fair distance (60 cm to 100 cm) from the Kinect



2. Run the program to output the distance that is captured between the Kinect and the flat object.
3. Use a measuring tape to measure the distance between the Kinect and the object to determine where the centre of the Kinect is

### Test Plans - Unit Testing for the Step Motor

Task: Test the precision of the step motor

Test Steps:

1. Mount a disc shaped paper onto the step motor, make sure that the centre of the disc is on the rod of the step motor
2. Draw a straight line from the centre to its edge
3. Place a 360° protractor underneath the step motor, ensuring that the 0° mark on the protractor matches the straight line drawn on the disc
4. Run the program to make the disc rotate at a designated degree
5. Once finished, check to see where the line on disc moved matches the value that was the input

Task: Test precision of the step motor with the Kinect mounted on it, to see if the precision will be affected with an object on top.

Task Steps:

1. Mount a disc shaped board onto the step motor, make sure that the centre of the disc is on the rod of the step motor
2. Draw a straight line from the centre to its edge
3. Mount the Kinect on the disc
4. Place a 360° protractor underneath the step motor, ensuring that the 0° mark on the protractor matches the straight line drawn on the disc
5. Run the program to make the disc rotate at a designated degree
6. Once finished, check to see where the line on disc moved matches the value that was the input

### Testing - Integration Testing

Task: Check the quality of the meshing when the different images are brought together.

Task Steps:

1. Post an image of a face on a flat wall at an angle such that only half of it can be seen by the Kinect
2. Run the Kinect and make it turn to the direction to get a full view of the image. This will capture two images, one with half of that face and second one with the full face.
3. Check in Meshlab to see if the images are brought together smoothly. Meaning are the two images on the same XY plane, or is one maybe little behind or in front of the other?

### Testing case-Graphical User Interface

For GUI testing, we need to consider the following in our test cases [8]:

- Domain size of operation
- The sequences of operation

Our project GUI only contains 10 operation sets (i.e. the domain size is 10). Unlike large commercial software, our GUI does not contain a lot of complexity, making the test process a lot simpler.

Thus, the test case for the GUI of the project will have:

- A simple automation to test each operation according to the specification checklist below.
- End-user manual testing will be done with scope of lecture class size of student with
  - White box testing
  - Black box testing

### General Principles

1. Every action that alters the user's data or application's settings can be undone
2. All application settings can be restored to their defaults with without the user having to remember what those defaults were.
3. The most frequently used functions are found at the top level of the menu structure.

### Graphical Elements Checklist

1. All multi-color graphical elements can be shown in monochrome only, where possible
2. All interactive GUI elements are easily distinguishable from static GUI elements
3. An option to hide non-essential graphics is provided

### Fonts and text

1. All labels have names that make sense when taken out of context.
2. No label names are used more than once in the same window.
3. Label positioning is consistent throughout the application.
4. Static text labels that identify other controls immediately precede those controls in the tab order.

### Color and contrast

1. Color is only used as an enhancement, and not as the only means to convey information or action.
2. The application supports all available high-contrast themes and settings
3. The software is not dependent on any particular high contrast themes or settings

### Timing checklist

1. The display or hiding of important information is triggered solely by movement of the mouse pointer.

## Documentation

1. All documentation is in an accessible format, with textual alternate descriptions provided for all figures.
2. The documentation includes a section that covers all the application's accessibility features.

## Test Plans - Power Benchmarks - Unit testing

Task: Measure Battery Life for QuickScan battery pack – Idle Mode

Test Steps:

1. Turn on device – ensuring battery has been fully charged
2. Do not initiate a scan process but keep the power connected to all modules
3. Measure the battery life for this idle mode

Task: Measure Battery Life for QuickScan battery pack – Normal Mode

Test Steps:

1. Turn on device – ensuring battery has been fully charged
2. Initiate first scan
3. Once scan is completed wait for 10 minutes before starting next scan
4. Perform steps 2-3 until battery is at zero capacity
5. Measure the battery life for this normal mode

Task: Measure Battery Life for QuickScan battery pack – Continuous Mode

Test Steps:

1. Turn on device – ensuring battery has been fully charged
2. Initiate first scan
3. Once scan is complete proceed to next scan with a minimal wait time
4. Perform steps 2-3 until battery is at zero capacity
5. Measure the battery life for this continuous mode



## 9. Environmental Considerations

### 9.1 Project Disposal

Dimension Technologies is committed to maintaining a high standard of quality and professional integrity. This includes considerations to environmental consequences resulting from the QuickScan module. Specifically, at the completion of our ENSC 440 project our device will be disposed in such a manner that maintains environmental standards and proper disposal practices. We will distribute the parts from the QuickScan module to our various team members for their own personal interests. Any leftover parts will be correctly disposed meaning that electronic parts will be disposed in electronic recycling center.

### 9.2 Product Disposal

Similar to our project disposal, in the field, customers can dispose of our QuickScan product at any electronic recycling center. Our product does not contain environmentally hazardous materials, thus standard electronic disposal techniques are sufficient.

## 10. Conclusion

This document has extensively explained our design process and methodologies for the QuickScan device. Each of our subcomponents has been thoroughly explained in addition to justifying our selection of such parts. A working prototype is expected to be functional by the first week of December.

## 11. References

- [1] Wikipedia Contributors. “Kinect” Internet: <http://en.wikipedia.org/wiki/Kinect> [Nov 8<sup>th</sup> 2012]
- [2] Microsoft Corporation. “Kinect” Internet: <http://www.xbox.com/en-CA/Kinect> [Nov 8<sup>th</sup> 2012]
- [3] Arduino Corporation “Arduino Nano”  
Internet: <http://www.arduino.cc/en/Main/ArduinoBoardNano> [Nov 8<sup>th</sup> 2012]
- [4] “Stepper Motor with Cable” Internet: <https://www.sparkfun.com/products/9238> [Nov 8<sup>th</sup> 2012]
- [5] “EasyDriver Motor Driver” Internet: <https://www.sparkfun.com/products/10267> [Nov 8<sup>th</sup> 2012]
- [6] “Triple Axis Magnetometer Breakout HMC5883L”  
Internet: <https://www.sparkfun.com/products/10530> [Nov 8<sup>th</sup> 2012]
- [7] “GUI Level Sequencing Testing”  
Internet: <http://www.lri.fr/~wolff/diss-proposals/GUI-levelSequenceTesting.pdf> [Nov 8<sup>th</sup> 2012]
- [8] “GUI Testing and Coverage Analysis” Internet: <http://www.ijicic.org/ijicic-09-1178.pdf>  
[Nov 8<sup>th</sup> 2012]
- [9] “OpenNI” Internet: [www.openni.org](http://www.openni.org) [Nov 8<sup>th</sup> 2012]
- [10] “Processing” Internet: [www.processing.org](http://www.processing.org) [Nov 8<sup>th</sup> 2012]
- [11] “ControlP5 Library” Internet: <http://www.sojamo.de/libraries/controlP5/> [Nov 8<sup>th</sup> 2012]
- [12] “Microsoft Kinect Teardown”  
Internet: <http://www.ifixit.com/Teardown/Microsoft+Kinect+Teardown/4066/2> [Nov 8<sup>th</sup> 2012]
- [13] “How Kinect Works Prime Sense”  
Internet: <http://ntuzhchen.blogspot.ca/2010/12/how-kinect-works-prime-sense.html>  
[Nov 8<sup>th</sup> 2012]
- [14] “How does the Microsoft Kinect work from a Technology Standpoint”  
Internet: <http://www.quora.com/How-does-Microsofts-Kinect-work-from-a-technology-standpoint> [Nov 8<sup>th</sup> 2012]
- [15] “Arduino Reference” Internet: <http://arduino.cc/en/Reference/HomePage> [Nov 8<sup>th</sup> 2012]

- [16] “Arduino Main Board” Internet: <https://www.sparkfun.com/products/666?> [Nov 8<sup>th</sup> 2012]
- [17] “Tutorial: I<sup>2</sup>C and Processing”  
Internet: <http://www.jeremyblum.com/2011/02/13/arduino-tutorial-7-i2c-and-processing/>  
[Nov 8<sup>th</sup> 2012]
- [18] “Wire Reference” Internet: <http://www.arduino.cc/en/Reference/Wire> [Nov 8<sup>th</sup> 2012]
- [19] “Universal Mounting Hub” Internet: <https://www.sparkfun.com/products/10006> [Nov 8<sup>th</sup> 2012]
- [20] “Stepper Motor” Internet: [http://en.wikipedia.org/wiki/Stepper\\_motor](http://en.wikipedia.org/wiki/Stepper_motor) [Nov 8<sup>th</sup> 2012]
- [21] “Stepper Motor” Internet: [http://www.omega.ca/prodinfo/stepper\\_motors.html](http://www.omega.ca/prodinfo/stepper_motors.html) [Nov 8<sup>th</sup> 2012]
- [22] “Basics of Motion Control”  
Internet: <http://www.orientalmotor.com/technology/articles/2phase-v-5phase.html>  
[Nov 8<sup>th</sup> 2012]
- [23] “EasyDriver” Internet: <http://schmalzhaus.com/EasyDriver/> [Nov 8<sup>th</sup> 2012]
- [24] “EasyDriver” Internet: <http://bildr.org/2011/06/easydriver/> [Nov 8<sup>th</sup> 2012]
- [25] “HMC5883L”  
Internet: <http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Magneto/HMC5883L-FDS.pdf> [Nov 8<sup>th</sup> 2012]
- [26] “Arduino” Internet: [http://bildr.org/2012/02/hmc5883\\_arduino/](http://bildr.org/2012/02/hmc5883_arduino/) [Nov 8<sup>th</sup> 2012]
- [27] “Tutorial: HMC5883L and Arduino”  
Internet: <https://www.loveelectronics.co.uk/Tutorials/8/hmc5883-tutorial-and-arduino-library>  
[Nov 8<sup>th</sup> 2012]
- [28] “Tripod Stand” Internet: <http://grabcad.com/library/tripod-stand--1> [Nov 8<sup>th</sup> 2012]
- [29] “Kinect” Internet: <http://www.chiefdelphi.com/forums/showthread.php?t=102002>  
[Nov 8<sup>th</sup> 2012]
- [30] E Melgar, C Diez. *Arduino and Kinect Projects*. Technology in Action, 2012
- [31] Cover Photo: “Lobby Room” Internet: <http://stockfresh.com/image/402224/lobby-room>  
[Nov 8<sup>th</sup> 2012]

Simon Fraser University,  
8888 University Dr.  
Burnaby, BC Canada  
Email: [rmt3@sfu.ca](mailto:rmt3@sfu.ca)

