Letter of Transmittal

John Schlichther

Electronics Engineering Technology - Industrial Automation

2000 Talbot Rd

Windsor, ON N9G 3C3

March 26th, 2020

Dear John Schlichther,

Articulating Robotic Arm:

A Proposal

By

Mathew Pellarin

Submitted on:

March 26th, 2020

Prepared for:

John Schlichther, Teacher

St Clair College, Windsor, Ontario

Declaration

I hereby declare that the project work entitled “Articulating Robotic Arm” submitted to St. Clair College of Applied Arts and Technology, is a record of an original work done by me under the guidance of Mr. John Schlichther, Professor of Electronics Engineering Technology, St. Clair College of Applied Arts and Technology, and this project work is submitted in the partial fulfillment of the requirements for the award of the diploma of Electronics Engineering Technology – Industrial Automation. The results embodied in this thesis have not been submitted to any other College or Institute for the award of any degree or diploma.

Mathew Pellarin

Mathew Pellarin

March 26th, 2020

Table of Contents

Contents

List of Figures

List of Tables

Acknowledgements

Firstly, I would like to say thanks towards my peers and colleagues. They have helped greatly with pointing me into the right direction with their constructive criticism and feedback. Apart from suggestions, they have also helped me with providing me with certain components instead of needing to purchase my own.

Next, I would like to express great gratitude toward my mother for providing a space to work and helping me along this project. They have provided the utmost encouragement and motivation to make sure I complete this final year project to my fullest potential.

Abstract

This final year project presents an Articulating Robotic Arm that aims to create a robotic assistant that can help reduce workplace injuries and help reduce workload on employees. With industries becoming more demanding with production speed and timelines, continues strain is put on the employees. With robots in the workplace, they can help employees get their job done with more efficiency and more accuracy. The electromagnet provides a safe and source way to pick up the part required with great holding force. For parts that are not finished, this provides and simple and effective way to suspend them in the air. The two main disadvantages of using an electromagnet are that the part may be scratched, or damages may occur, and the part must be made of metal.

Introduction

With products becoming more complex and industries trying to keep up with quotas and minimizing downtime, machines are an amazing way to make sure these requirements are met. With robots more involved in the workplace, an increase in efficiency and quality can be seen.

Robots can complete certain tasks many times faster and with greater accuracy than humans. Where humans may take over a minute to screw one screw, robots can have the task complete in mere seconds. Also, in welding, for a human to get a near perfect weld may take significant time where a machine should be able to get near perfect in a fraction of that time with. With reduced time also implies reduced labor cost and a lower cost per product.

With reduced cost per product, profitability can be also be increased. The upfront cost of the robot may seem significant, but they will soon pay for themselves bringing in more profit. They can work twenty-four hours a day, every day of the week at 100% efficiency. With no breaks for holidays or unexpected absences these machines can run every day for many years or even decades.

Background Theory

This machine has many intricate circuit boards with many components and parts involved. Many devices such as resistors, optocouplers, LEDs, buck converters and much more have a specific faction to make sure the entire machine works.

Resistors

Resistors are passive components to restrict the flow of current of a given circuit. This is achieved by changing electrical power to heat to reduce current flow. Resistors have many purposes in electronics. They can be used as voltage dividers, current limiters, or to make sure the floating voltage in a circuit is brought either up to a constant voltage or down to zero volts. The measurement across the resistor can be expressed in ohms. With this value, calculations can be done to determine how much current is restricted across the resistor. The expression can be displayed as resistance is equal to voltage divided by current. If there were no resistance in each circuit, a complete short would be created. Since there is no resistance, there is no restriction to current flow and electrical flow would be next to infinite. This would cause a surge of power and many components including power supplies may self-destruct. Resistance is also directly proportional to voltage. With more resistance comes more of a voltage drop across the resistor.

$$R = \frac{V}{I}$$

Figure 1-1: Resistance Equation

Figure 1-2: Resistor

Light Emitting Diodes

Light emitting diodes (also known as LED) is a semiconductor that produces a light as current flows through it. LEDs are significantly more efficient than traditional incandescent light due to less heat being generated and less power requirements. LEDs are usually smaller than most traditional lights saving space. They have a long-life span averaging over 5 years with a constant usage twenty-four hours a day. Though since LEDs are part of the diode family, power may only travel one way.

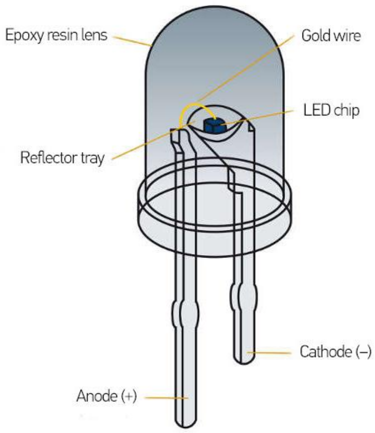
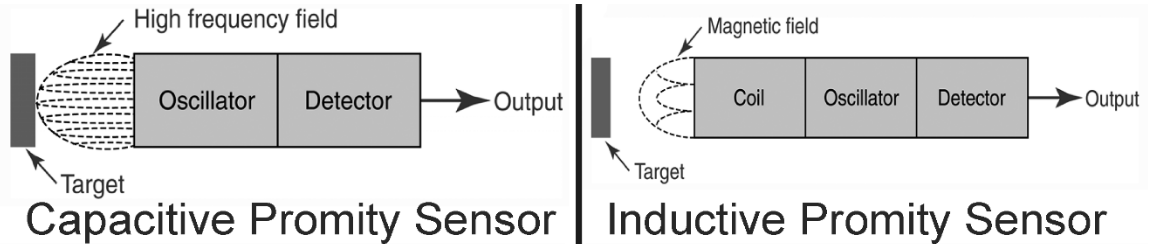


Figure 2-1: LED Close-up

Proximity Sensors

Proximity sensors are used to detect the presents of something near it. The product does not have to be touching the proximity sensor, instead it can be close and still detect an accurate reading. There are two types of proximity sensors used in industry. There is a capacitive proximity sensor and an inductive proximity sensor. Both are created similarly but detect different items. near the device were an inductive proximity sensor only detects the presents of metal objects. Inductive proximity sensors are very useful as well because they can also detect metal behind objects such as the conveyor or cardboard. The inductive proximity sensor has a coil of winding inside with an oscillator and amplifier. The oscillator is used to maintain a constant frequency through the coil. Since there is an oscillation in the coil, a magnetic field is generated. If the oscillations amplitude drops, this means a metal part is detected. The capacitive proximity sensors have a capacitor plate instead a coil. This induces an electric field. As the oscillation frequency becomes higher, the part present is detected.



Capacitive Promity Sensor

Inductive Promity Sensor

Figure 3-1: Proximity sensors

Optocouplers

Optocouplers are a great way to isolate different circuits or to interface with different voltages. Optocouplers are a non-mechanical type of integrated circuits that have a light emitting diode on one side and the other side a phototransistor. When the light emitting diode is turned on, the phototransistor sees light and allows current to passthrough, like a switch. When the light emitting diode is turned off, the phototransistor does not allow current to flow. When the light emitting diode is not fully on, the phototransistor will allow only a percentage of the current to flow also known as current transfer ratio (CTR). The current transfer ratio can be calculated by dividing the input current of the led by the input current of the transistor.

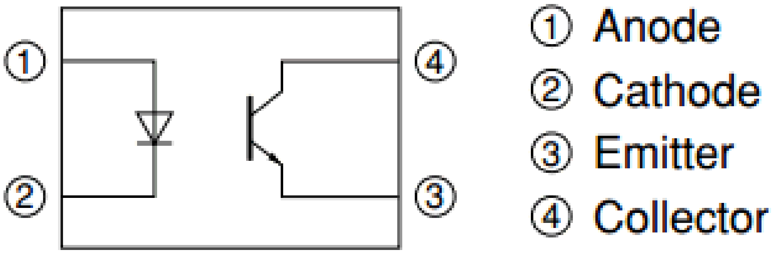


Figure 4-1: Optocoupler electrical drawing

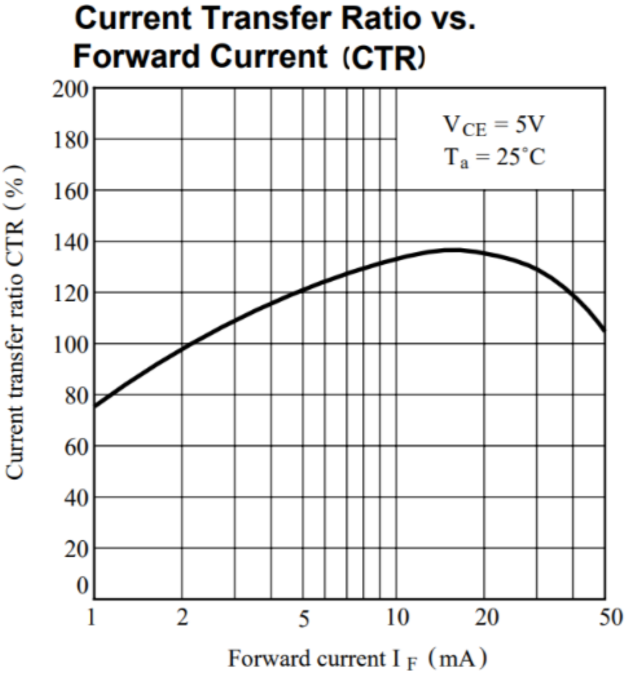


Figure 4-2: Optocoupler CTR of PC817c

Servos

Servos are precision controlled rotational actuator. They can rotate to the angle specified and lock into place once there. Servos consist three main components, a DC motor, potentiometer and a gearbox. The gear box is connected to both the DC motor and potentiometer. As the DC motor spins, the potentiometer is turned allowing the servo to know precisely where it is located and when to stop moving. A pulse width modulation (PWM) signal input to the servos tells the servo where to move.

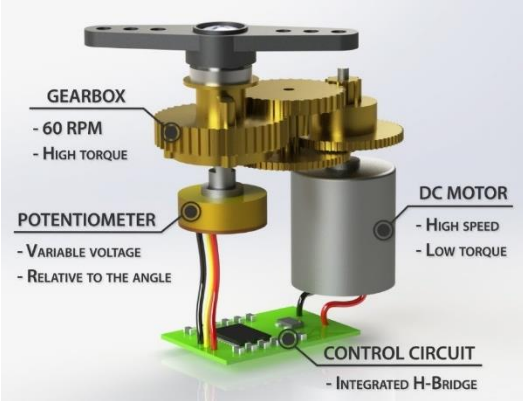


Figure 5-1: Inside RC Servo

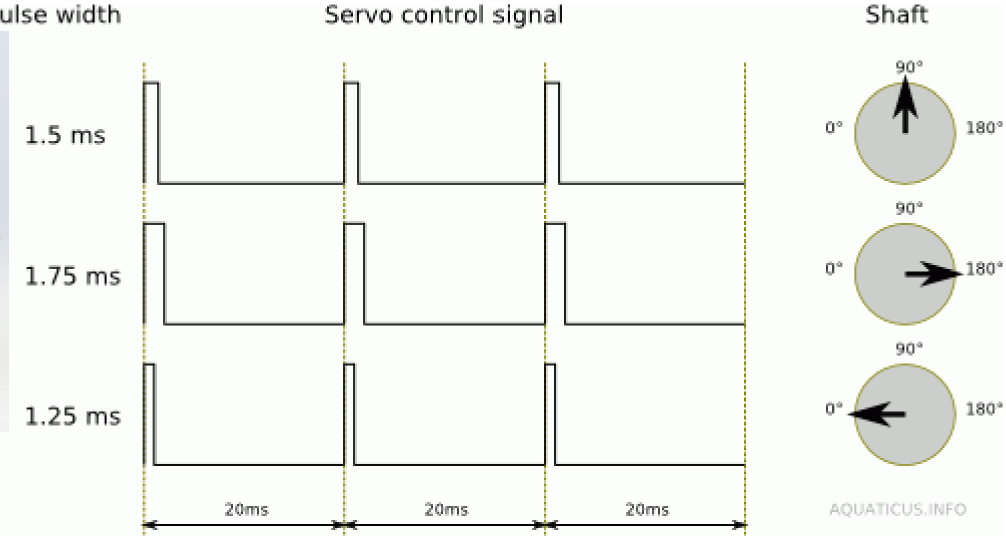


Figure 5-2: PWM timing of RC Servo

Stepper Motor

Stepper motors are also precision controlled. Though with these motors, they can rotate 360 degrees with great accuracy. Inside a stepper motor has great complexity. On an average stepper motor, the magnetic rotor has 50 teeth with the stator having 48. These are in precise locations to off kilter some teeth. The stator has two sets of electromagnets to be energized and shift the rotor around. When one set is energized, the rotor is rotated to align with the offset teeth. When the next set is energized, the rotor then turns again to match the next set of teeth. This is continued till an exact direction is needed. When stepper motors are paired with a stepper motor driver, they can be easy to control with great advantages. Stepper motors have an amazing holding torque when left on.

When paired with a basic driver, stepper motors have an accuracy of 1.8 degrees per pulse. When pair with an advance driver, they can have an accuracy of less the 0.01 degrees per pulse. Though the major drawback of these motors is that the position of the stepper motor is unknown unless tracked by the microcontroller.

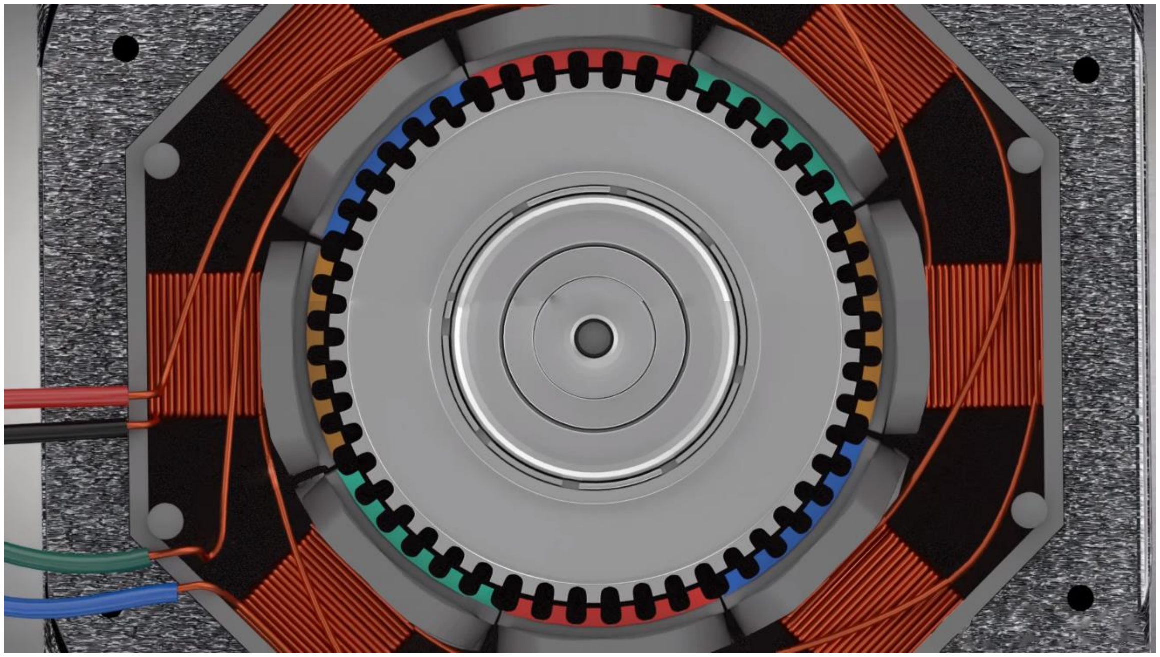


Figure 6-1: Inside a Stepper Motor

Relays

Relays are a device like optocouplers but are the mechanical versions. Relays consisted of an electromagnet and an armature with contacts. When the electromagnet is energized, the armature pivots usually away from the electromagnet touching two contacts together. This allows for circuit isolation, one side to energize the coil and the other through the contact pads. This mechanical device allows for much more power to be switched on and off electrically.

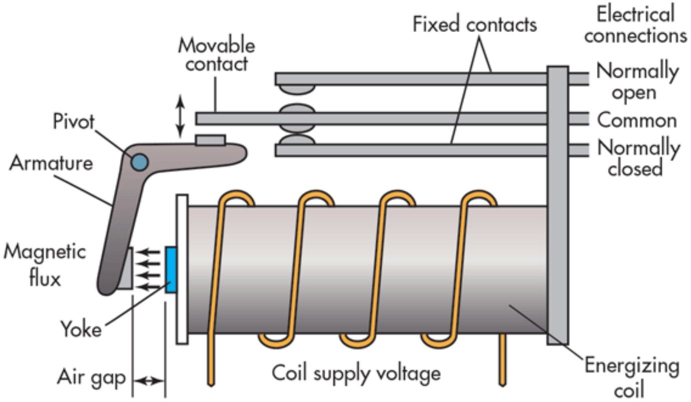


Figure 7-1: Inside a relay

Fuses

A fuse is an electrical safety device that self-destructs to protect the rest of the circuitry. A fuse consists of an enclosure with a small metal strip inside. When this metal strip heats up due to too much current flow, the wire melts and severs the connection. This save many sensitive devices that cannot take the surge of power.

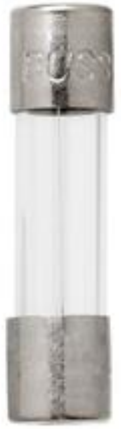


Figure 8-1: Glass fuse

System Design

The design stage needed to be well thought-out before the construction of the articulating robotic arm was built. All resistor values, motor sizes, voltage difference control and much more had to be carefully planned.

When starting the design process, major electrical components were picked. Servos would be used to give the arm motion. Electromagnet was the tip of the arm to pick up the parts. And stepper motor was going to be used for the conveyor belt. This would be all controlled by an Arduino Mega. Next, mechanical draws were created.

Mechanical Design

The Arm

When designing the mechanical portion, the first thing was to decide what material was going to be used to create the arm. The material chosen was 3D printable Polylactic acid (PLA). This way it would be easy to create parts from scratch. Polylactic acid is quite strong of a martial and shouldn't break under the conditions the robotic arm will be placed under. The servos will be pressure fitted into the cutouts as well as screwed down to make sure the best durability can be reached (Refer to appendix 2-1). The size of the arm was limited to the size of the 3D printer bed. Each part had to be less then 200mm by 200mm. The arm of the robot is 160mm and is broken into two pieces (Refer to appendix 2-1). Two servos will be mounted into each piece and give the arm more flexibility to move. The head of the arm is about 55mm tall and large enough to enclose the electromagnet inside the housing (Refer to appendix 2-3). The base consists of two servos. One allowing the arm to rotate itself around it to place the part in desired area and the other giving the arm more range and motion. The base is 120mm wide and 140mm tall (Refer to appendix 2-2). This give the arm a great amount of rigidity and durability enabling itself to pick up the part needed.

The Conveyor

The Conveyor design was the next big task on hand. The material chosen for the base of the conveyor was aluminum extrusion (Refer to appendix 2-5). This way the conveyor can be easily adjustable and shifted around for the correct tension of the belt. Also, it would be easy to mount the proximity sensors to this. The belt would be made of PVC and span about 75cm long with a width of 5cm. The rollers would be also PVC pipe attached to with some bearings (Refer to appendix 2-6). A threaded rod will be used to hold the rollers in place and coupled with the stepper motor on the end of one side. The inductive proximity sensors will be mounted under the conveyor belt to detect when the part comes down the conveyor.

Electrical Design

Microcontroller

The microcontroller chosen was the Arduino Mega. This microcontroller has a total of fifty-two digital inputs or outputs. This is suitable for the thirty-two I/O that would be required (Refer to appendix 2-3, 2-4, 2-5). The voltage of the I/O is five voltages. With these low voltages, it is easy and safe to interface with many other devices. The Arduino Mega also is a very cheap and reliable controller. This way if anything disasters were to happen while the construction phase was happening, a replacement would be readily available while staying on the budget. The Arduino is also quite fast with running at sixteen megahertz the entire code will be able to execute and not worry about delays from the processor. The Arduino Mega also has the most storage out of all the Arduinos. With two hundred fifty-six kilobytes of flash memory, there should be no chance of running out of storage.

LED Driver

The LED driver takes the five-volt signal from the Arduino and steps the voltage up to twelve volts. This circuit is also designed to isolate the two voltages to protect the Arduino. This circuit was designed so that when the Arduino has turned on the output, the onboard LED turns on and the Optocoupler also turns on. The reason for the three hundred thirty-ohm resistor is to current limit so that the LED or Optocoupler do not blow (Refer to appendix 2-6). The resistor in the circuit only allows four and a half milliamps of current through both the LED and optocoupler. This current is derived from taking the input voltage of five volts and subtracting the voltage drop of the LED (2.3 volts) and the optocoupler (1.2 volts) resulting in a value of one and a half volts. Divide this number by the resistor value (330Ω) and resulting in a current of four and a half milliamps. This is enough current to turn the LED on and allowing the optocoupler to work in its saturation range and the CTR value to be over one hundred percent. All current will be able to be passed through on the other side allowing the indicator LEDs on the front panel to work at twelve volts.

Proximity Sensors

The proximity sensors that were chosen were the TL-W5MC1 (Refer to appendix 2-12. These proximity sensors are an NPN inducive proximity sensor with a detection distance of five millimeter. These support the twelve volts that will be provided and should detect metal through the two-millimeter conveyor belt.

Proximity Sensor Driver

The proximity sensor driver is similar to the LED but has in a reverse manner. The output from the proximity sensor is current limited by a five hundred eighty-ohm resistor (Refer to appendix 2-12). Since the output of the proximity sensors are twelve volts a higher resister is required than the LED driver. This resistor allows for fourteen milliamps to passthrough giving adequate current to the LED and optocoupler. The current can be calculated by taking the twelve volts supplied from the optocoupler and subtracting it from the LED (2.3 volts) and the optocoupler (1.2 volts) resulting in eight and a half volts. Dividing this value by the five hundred eighty-ohm resistor will give us the current value of fourteen milliamps. The output of the optocoupler is fully saturated and allows the output to be off. When the proximity sensor is turned off, the output is turned on. The ten-thousand-ohm resistor is a pull up resistor to make sure when the optocoupler is off, a small amount of current can pass through to give the Arduino a logic high.

Stepper Motor and Driver

The stepper motor selected was a Nema 23 high torque stepper motor. This motor can draw a maximum of four amps. This motor is driven by the TB6600 driver (Refer to appendix 2-2). This driver can supply the needed current to rotate the stepper motor and is able to easily interface with the Arduino. The stepper motor driver is set to quarter stepper for the stepper motor to slow the rotation down and increase accuracy.

Pulse per Full Step	RPM	Rad/s	cm/s
1	147	15.39	33.8664
2	73.5	7.697	16.9332
4	36.75	3.848	8.46659
8	18.38	1.924	4.2333
16	9.188	0.962	2.11665
32	4.594	0.481	1.05832

Table 1-1: Stepper Motor Conveyor Speeds

RC Servos

There were two different kinds of RC servos used for the robotic arm. Two were MG996R and the other two were Zoskay 25KG (Refer to appendix 2-11). The reason for this difference is because more torque was needed for the lower arm that the MG996R could not handle. The MG996R only support 12kg/cm oh holding torque where the Zoskay supported 25kg. This made the robotic arm move easily where it needed to go and be able to hold the part required. The RC servos can work on a voltage between 4.8 volts and 6.5 volts. To get the best performance, the 12 volt power supply was stepped down with a buck converter to 6.5 volts and that was used to drive all RC servo motors.

Electromagnet and Relay

To pick the part up the part, an electromagnet was used. This is a more reliable way to pick a part up rather than using a claw grip. The electromagnet chosen was the XRN-XP30x22 (Refer to appendix 2-2). This is a 12 volt electromagnet that can hold up to 10kg. This one was chosen because it was small and very light. At 30mm in diameter and 80 gram weight, the arm will not struggle to hold its own weight and a part at the same time. To control this with the Arduino, a relay was put into place. This way the Arduino is protected by the electromagnet 12 volt necessity to turn on.

Conclusion

The parts chosen for the articulating robotic arm have proven to be suitable for the application required. The robotic arm moves with no effort to its destination and the conveyor can withstand the load of the part required.

With the robotic arm, the RC servos used were a better choice than using the first idea which was stepper motors. Stepper motors would have had a tremendous amount of torque, but the weight would have also been a draw back. Stepper motors weight a lot more and in previous testing, they needed a more structural mechanical design. Also, stepper motors would need to be calibrated to determine there home every time the system was rebooted. When switching to RC servos, the MG996Rs proved they were almost strong enough for all motion. Once the part was picked up, the arm could no longer move. Replacing two of these RT servos with the Zoskay brand has verified that the arm now works effortlessly to pick the parts required.

The Conveyor system was originally supposed to have a DC brush motor. The MY68 was the model that was going to control the conveyor with a H-bridge controller. This motor would have worked well but the revolutions per minute were a little to high and the conveyor was turning to fast. A gear box or attaching a belt to drive the conveyor but more points of failure may have risen. Switching this motor to the donated stepper motor reduced the failure rate with more torque and easier speed control.

The electromagnet was the last change decided to the arm. Originally a claw style arm was decided. When testing this, the claw either didn’t have enough holding torque to hold the specified part and ended up snapping. After this happened, a decision was made to make the arm have an electromagnet to grab the given part. This also reduced weight and has a tremendously lower failure rate than the claw style.

References

Figure 1-1

https://slideplayer.com/slide/9834302/

Figure 1-2

https://www.electricalclassroom.com/electric-resistance-what-is-resistance/

Figure 2-1

https://www.hella.com/techworld/us/Technical/Automotive-lighting/LED-headlights-833/

Figure 3-1

Doug Watson's EET 361, Lecture (Week 10)

Figure 4-1

http://www.datasheetq.com/PIC817-doc-Sharp

Figure 4-2

https://pdf1.alldatasheet.com/datasheet-pdf/view/43376/SHARP/PC817C.html

Figure 5-1

https://www.youtube.com/watch?v=LXURLvga8bQ

Figure 5-2

http://www.aquaticus.info/pwm-servo/

Figure 6-1

https://www.youtube.com/watch?v=eyqwLiowZiU

Figure 7-1

https://www.machinedesign.com/mechanical-motion-systems/article/21834253/engineering-essentials-relays-and-contactors

Figure 8-1

https://www.elliottelectric.com/P/Item/BUS/GMA1A/

Bibliography

How To Mechatronics. “How Servo Motors Work & How To Control Servos using Arduino”. *Youtube*, March 18, 2018, www.youtube.com/watch?v=LXURLvga8bQ.

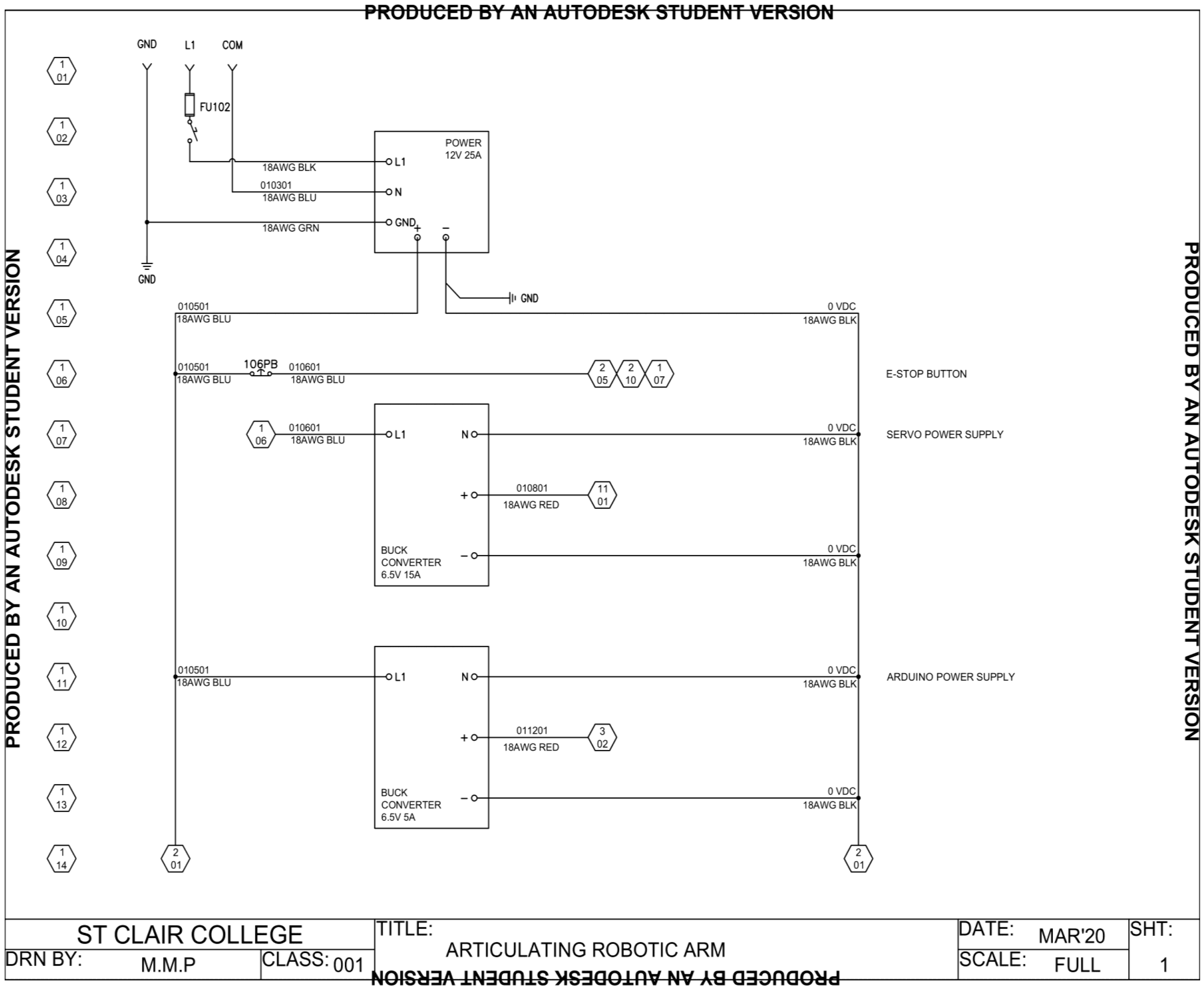
“Optocoupler Tutorial and Optocoupler Application.” *Basic Electronics Tutorials*, 16 May 2018, www.electronics-tutorials.ws/blog/optocoupler.html.

Learn Engineering. “How does a Stepper Motor work?”. *Youtube*, October 19, 2016, www.youtube.com/watch?v=eyqwLiowZiU.

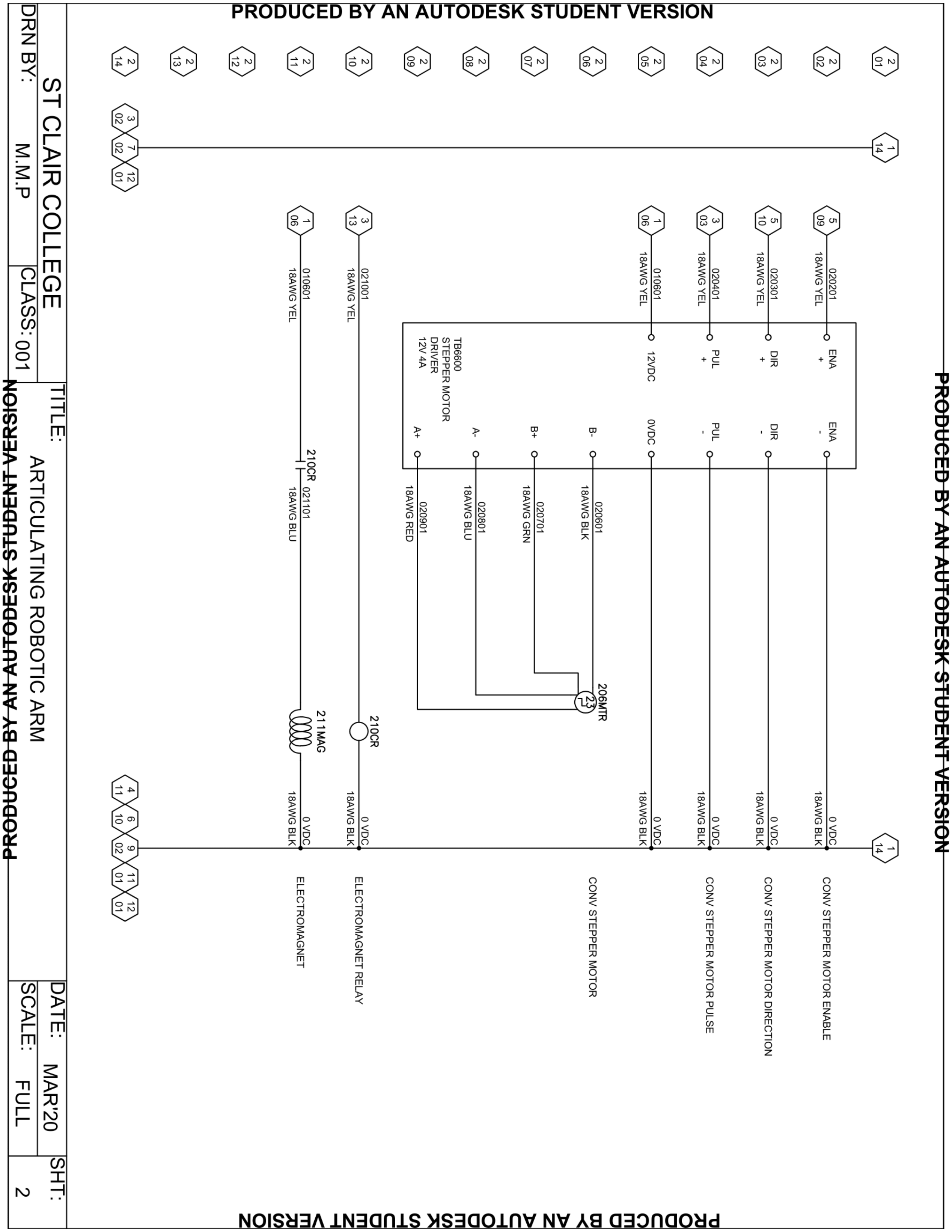
Appendices

Electrical Drawings

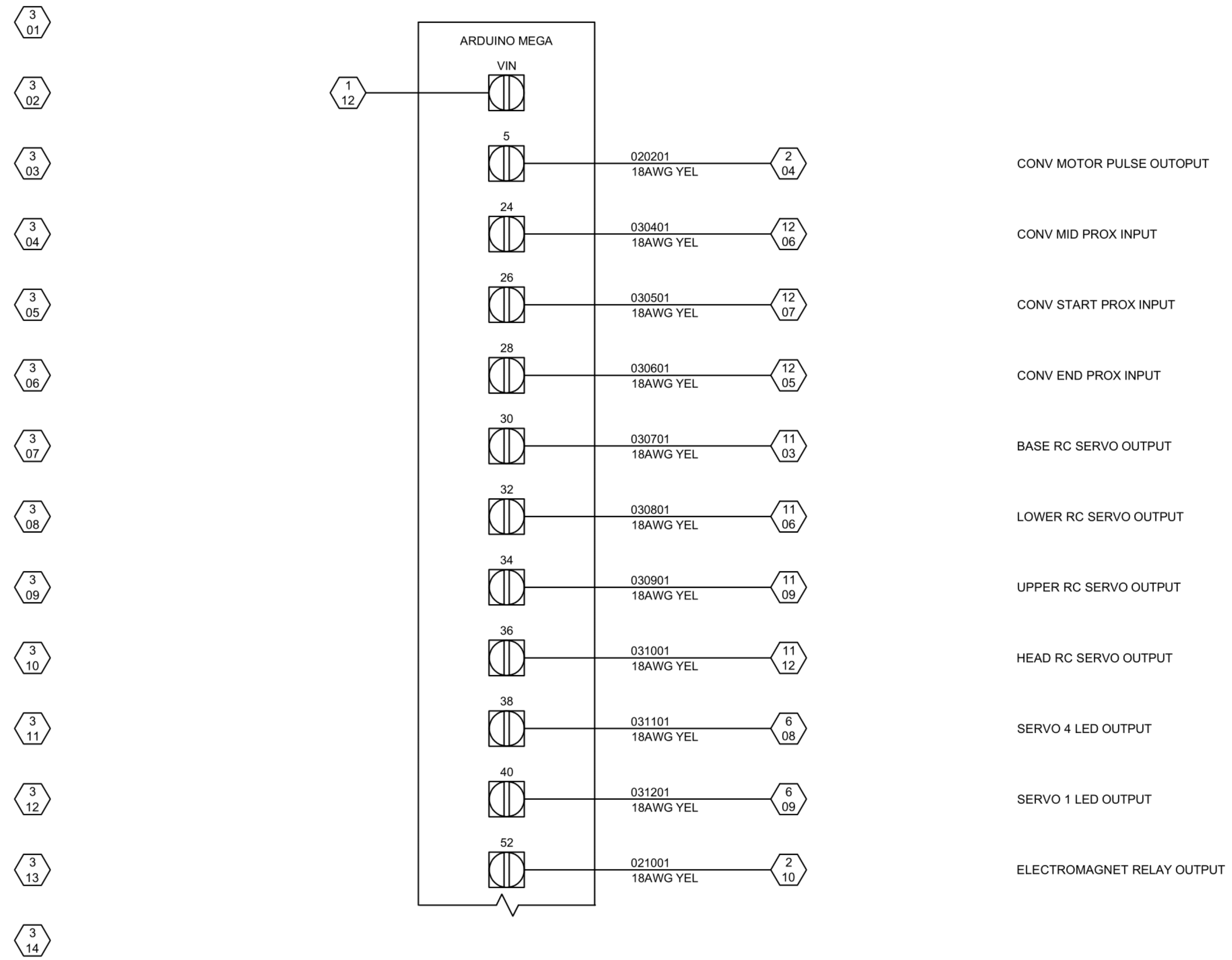
1-1



1-2

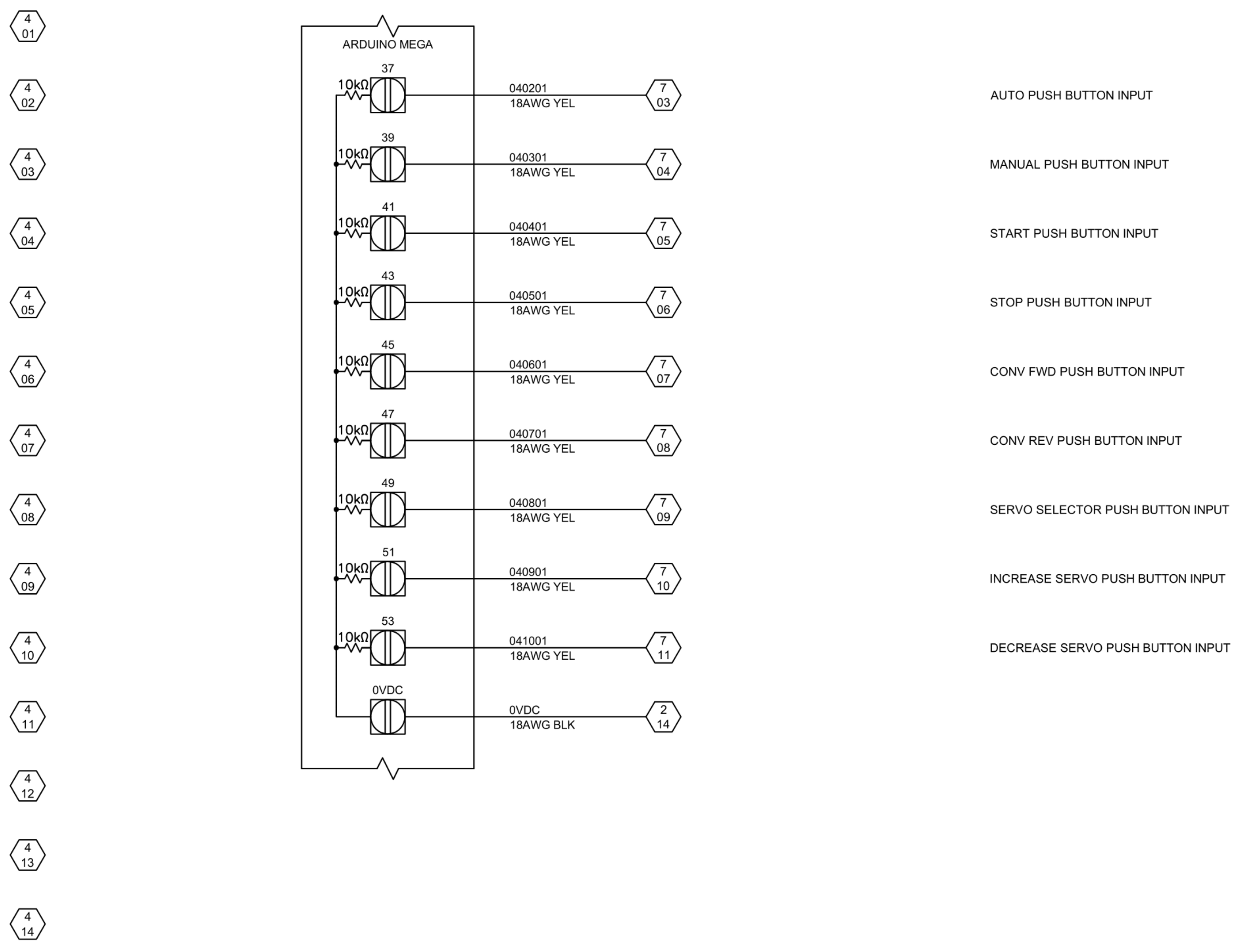

2 01
2 02
2 03
2 04
2 05
2 06
2 07
2 08
2 09
2 10
2 11
2 12
2 13
2 14
1 14
3 02
7 02
12 01
5 09
020201
18AWG YEL
ENA +
ENA -
0 VDC
18AWG BLK
CONV STEPPER MOTOR ENABLE
5 10
020301
18AWG YEL
DIR +
DIR -
CONV STEPPER MOTOR DIRECTION
3 03
020401
18AWG YEL
PUL +
PUL -
CONV STEPPER MOTOR PULSE
1 06
010601
18AWG YEL
12VDC
0VDC
CONV STEPPER MOTOR
TB6600
STEPPER MOTOR
DRIVER
12V 4A
B-
B+
A-
A+
020601
18AWG BLK
020701
18AWG GRN
020801
18AWG BLU
020901
18AWG RED
206MTR
23
3 13
021001
18AWG YEL
210CR
ELECTROMAGNET RELAY
1 06
010601
18AWG YEL
210CR
021101
18AWG BLU
211MAG
ELECTROMAGNET
4 11
6 10
9 02
11 01
12 01
ST CLAIR COLLEGE
DRN BY:
M.M.P
CLASS: 001
TITLE:
ARTICULATING ROBOTIC ARM
DATE:
MAR'20
SCALE:
FULL
SHT:
2

1-3



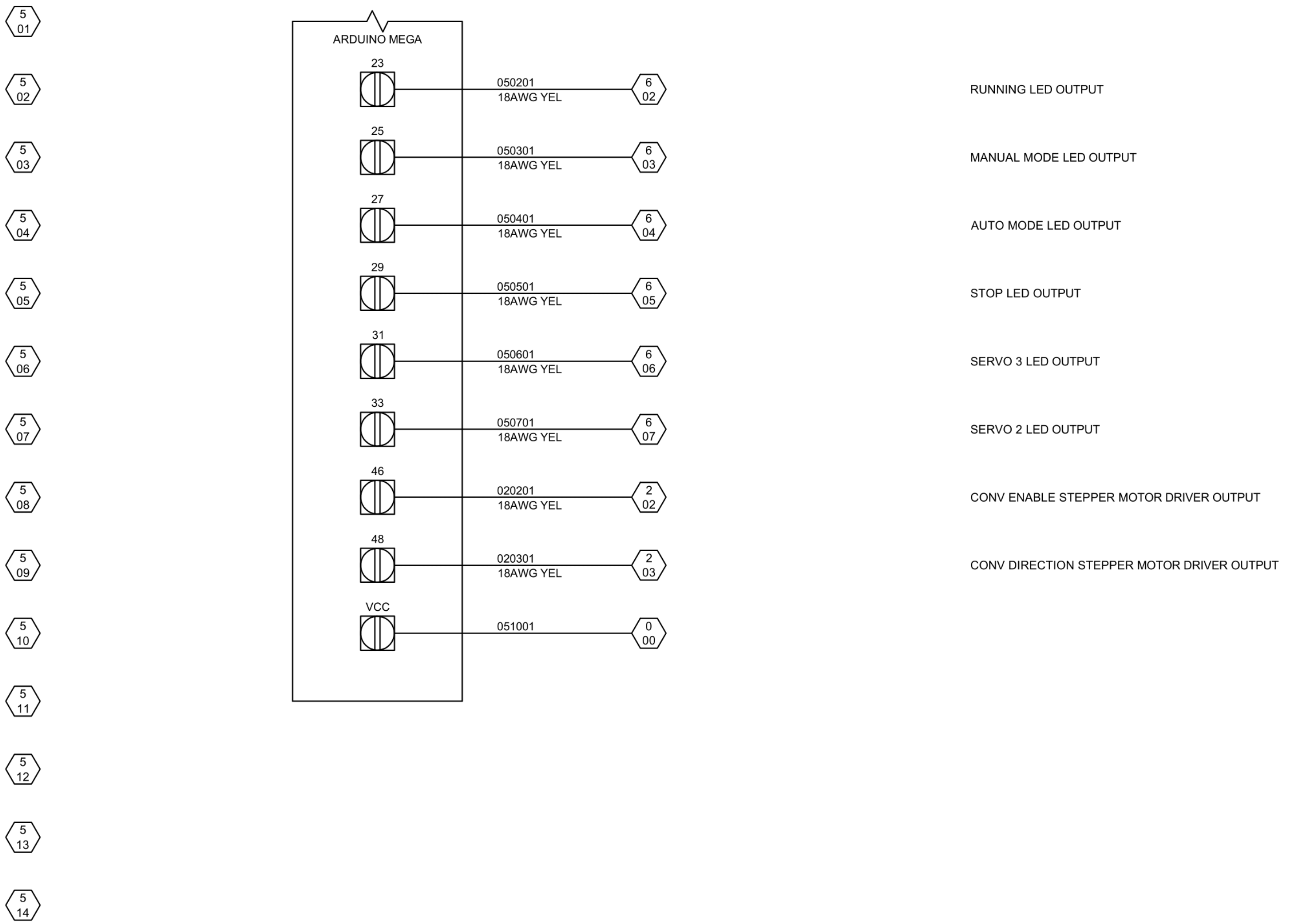
ST CLAIR COLLEGE	TITLE: ARTICULATING ROBOTIC ARM	DATE: MAR'20	SHT:
DRN BY: M.M.P · CLASS: 001		SCALE: FULL	3

1-4



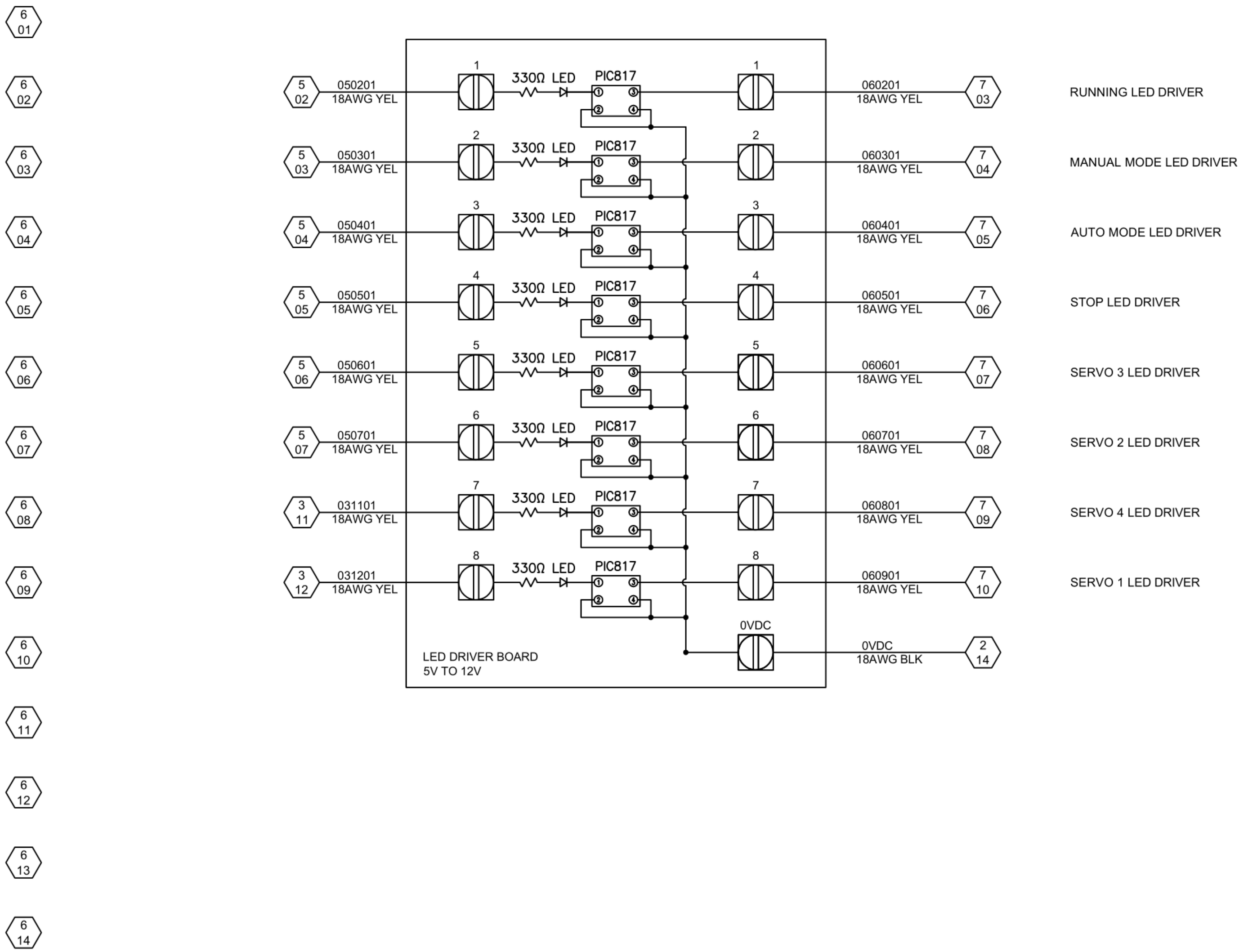
ST CLAIR COLLEGE		TITLE: ARTICULATING ROBOTIC ARM	DATE: MAR'20	SHT:
DRN BY: M.M.P	CLASS: 001		SCALE: FULL	4

1-5



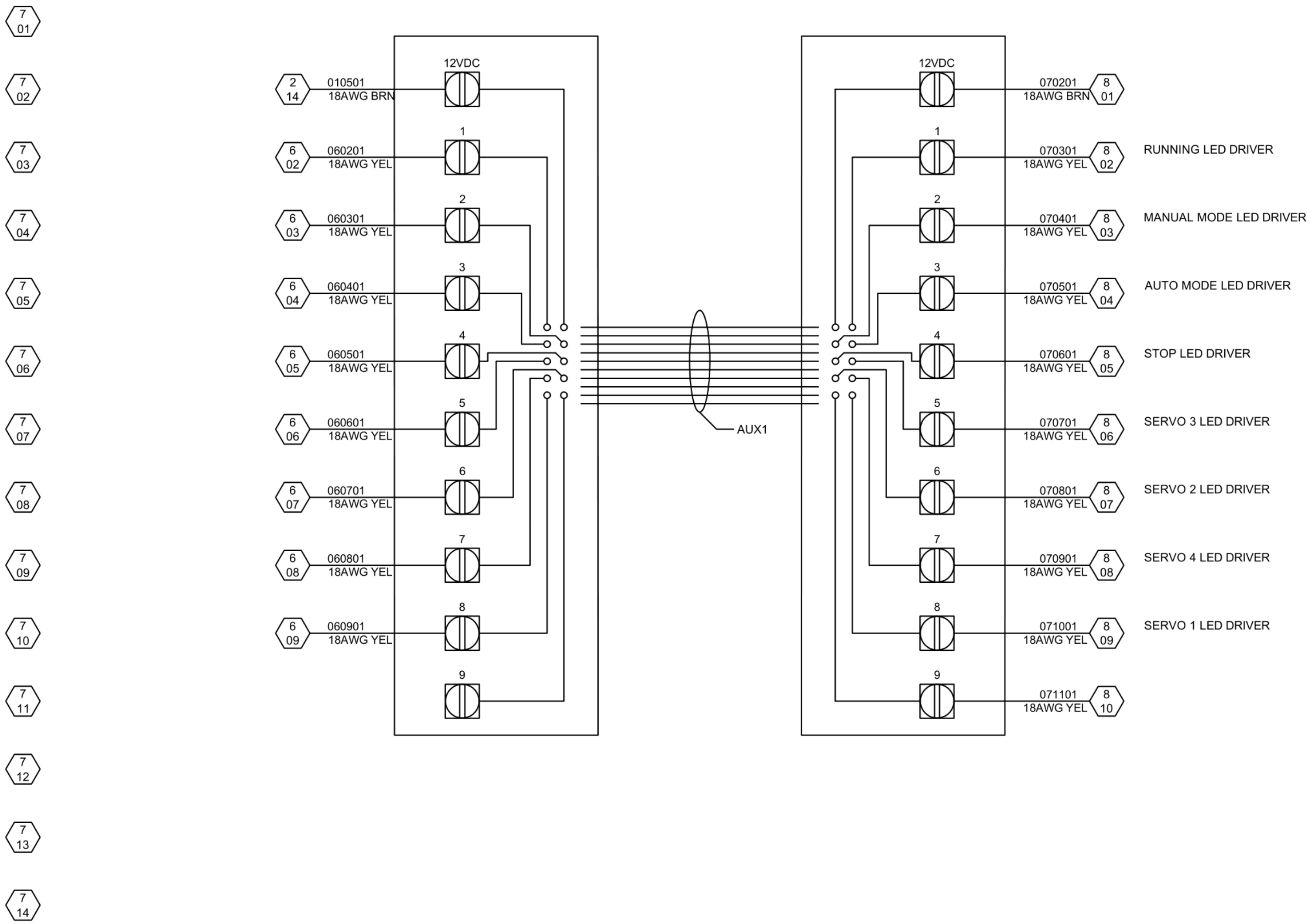
ST CLAIR COLLEGE		TITLE: ARTICULATING ROBOTIC ARM	DATE: MAR'20	SHT:
DRN BY: M.M.P	CLASS: 001		SCALE: FULL	5

1-6



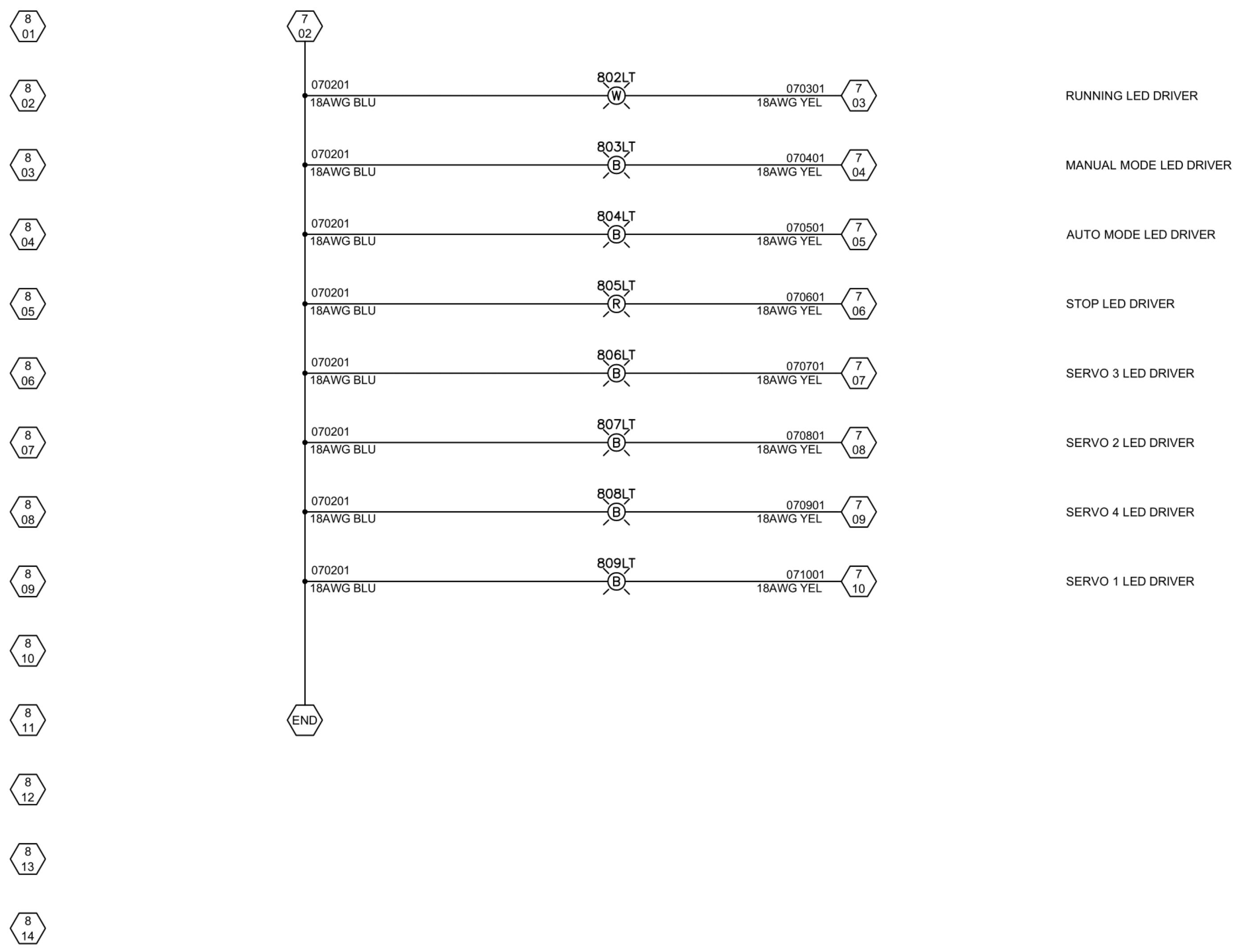
ST CLAIR COLLEGE		TITLE: ARTICULATING ROBOTIC ARM	DATE: MAR'20	SHT:
DRN BY: M.M.P	CLASS: 001		SCALE: FULL	6

1-7



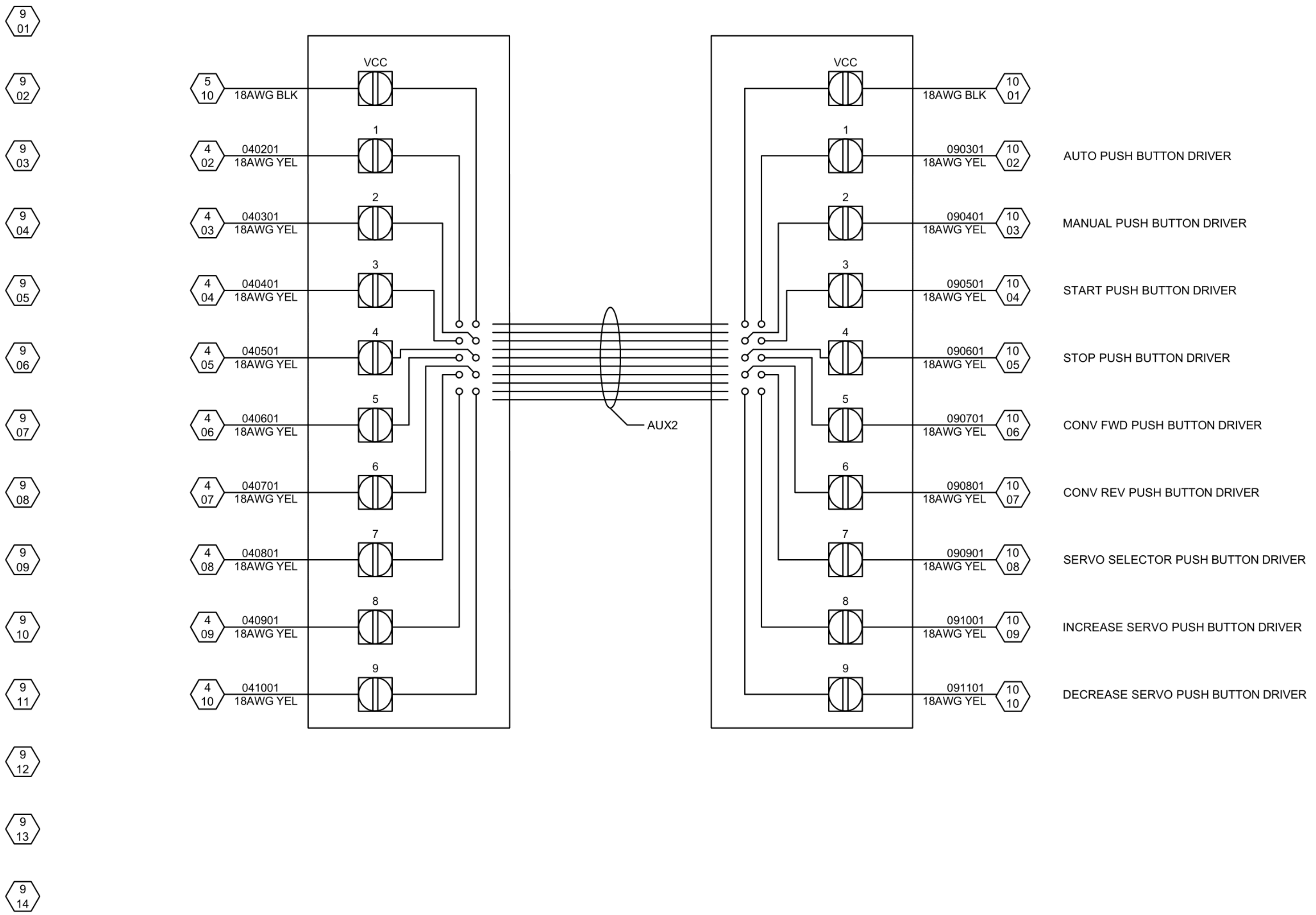
ST CLAIR COLLEGE	TITLE: ARTICULATING ROBOTIC ARM	DATE: MAR'20	SHT:
DRN BY: M.M.P	CLASS: 001	SCALE: FULL	7

1-8



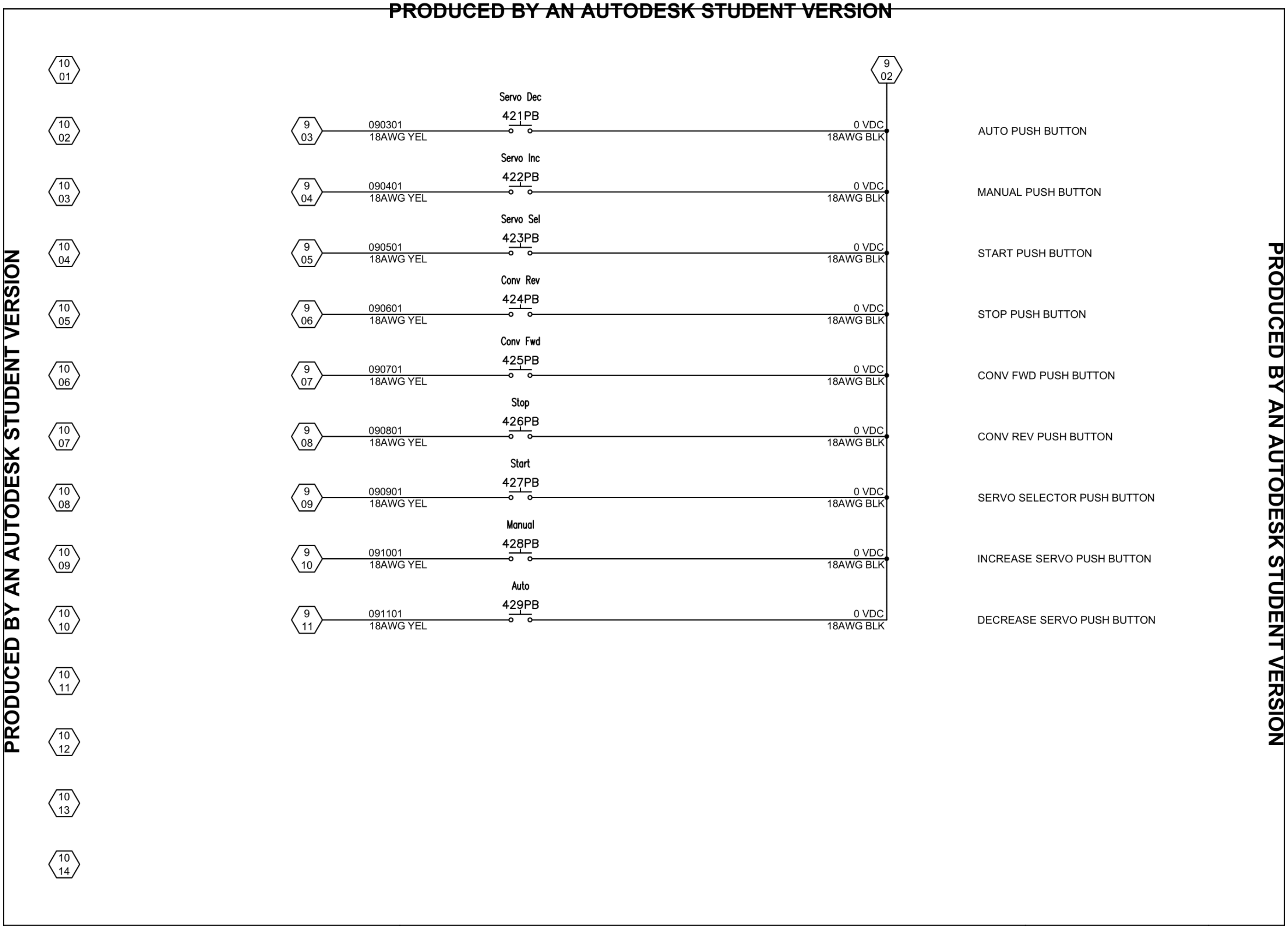
ST CLAIR COLLEGE		TITLE: ARTICULATING ROBOTIC ARM	DATE: MAR'20	SHT:
DRN BY: M.M.P	CLASS: 001		SCALE: FULL	8

1-9



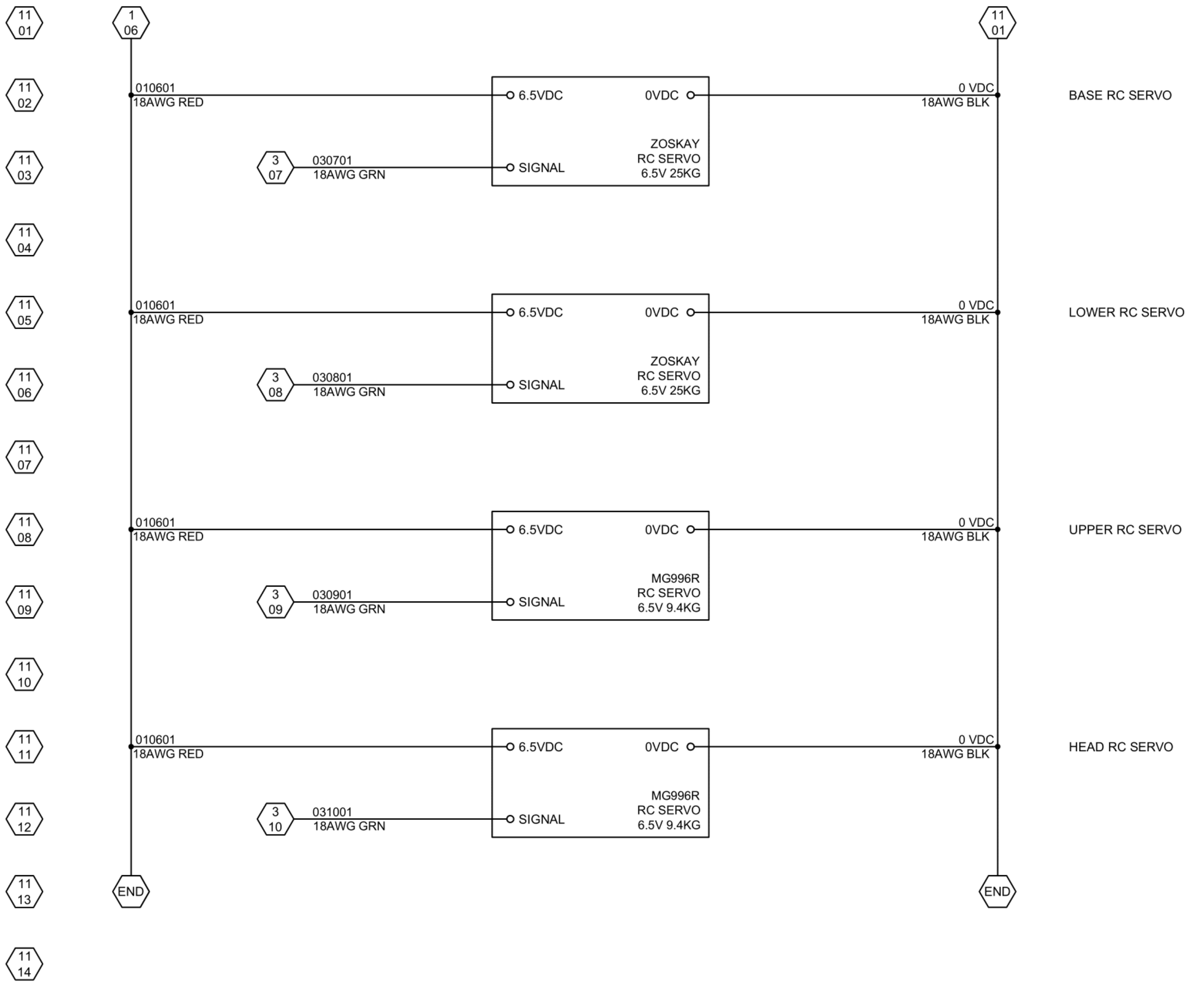
ST CLAIR COLLEGE	TITLE: ARTICULATING ROBOTIC ARM	DATE: MAR'20	SHT:
DRN BY: M.M.P CLASS: 001		SCALE: FULL	9

1-10



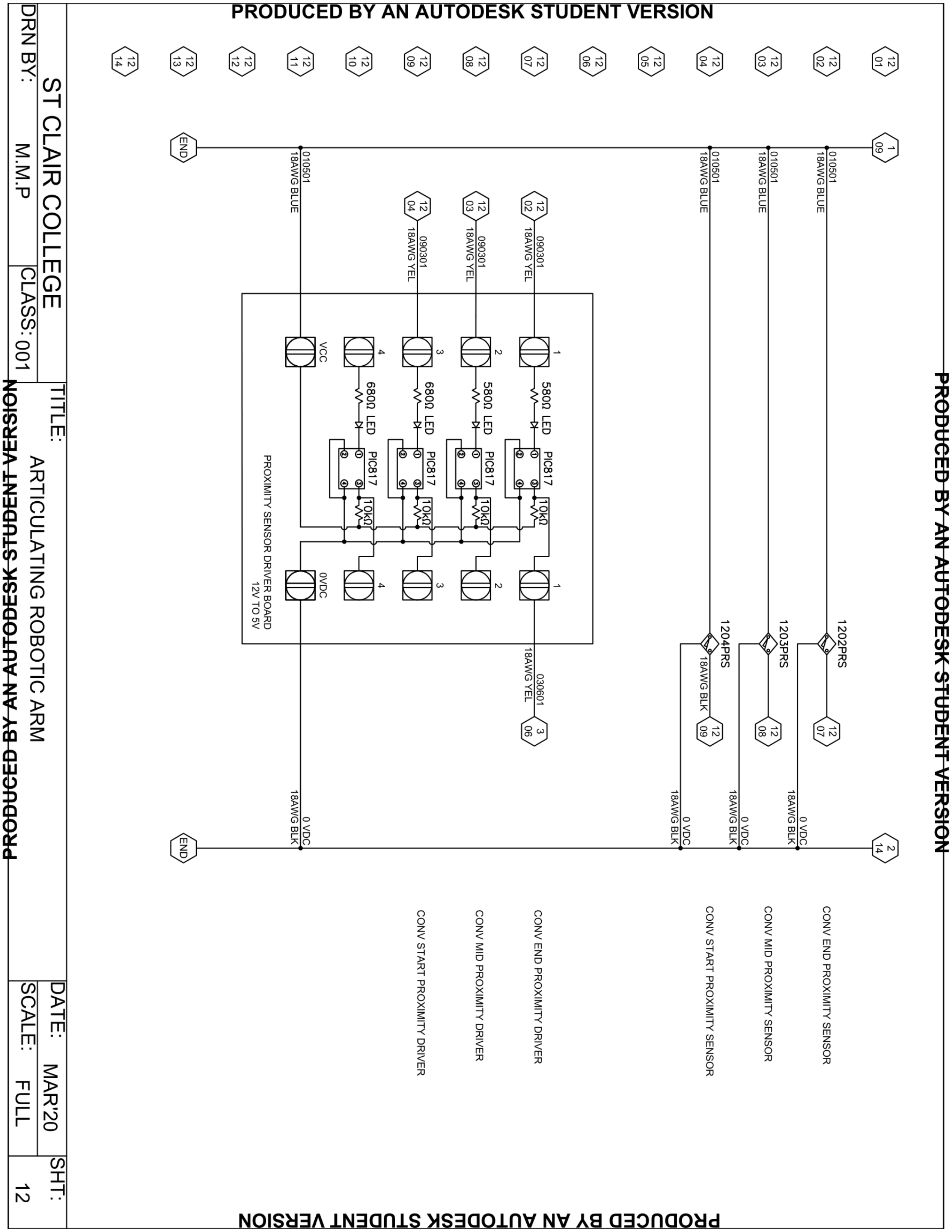
ST CLAIR COLLEGE	TITLE: ARTICULATING ROBOTIC ARM	DATE: MAR'20	SHT:
DRN BY: M.M.P | CLASS: 001		SCALE: FULL	10

1-11



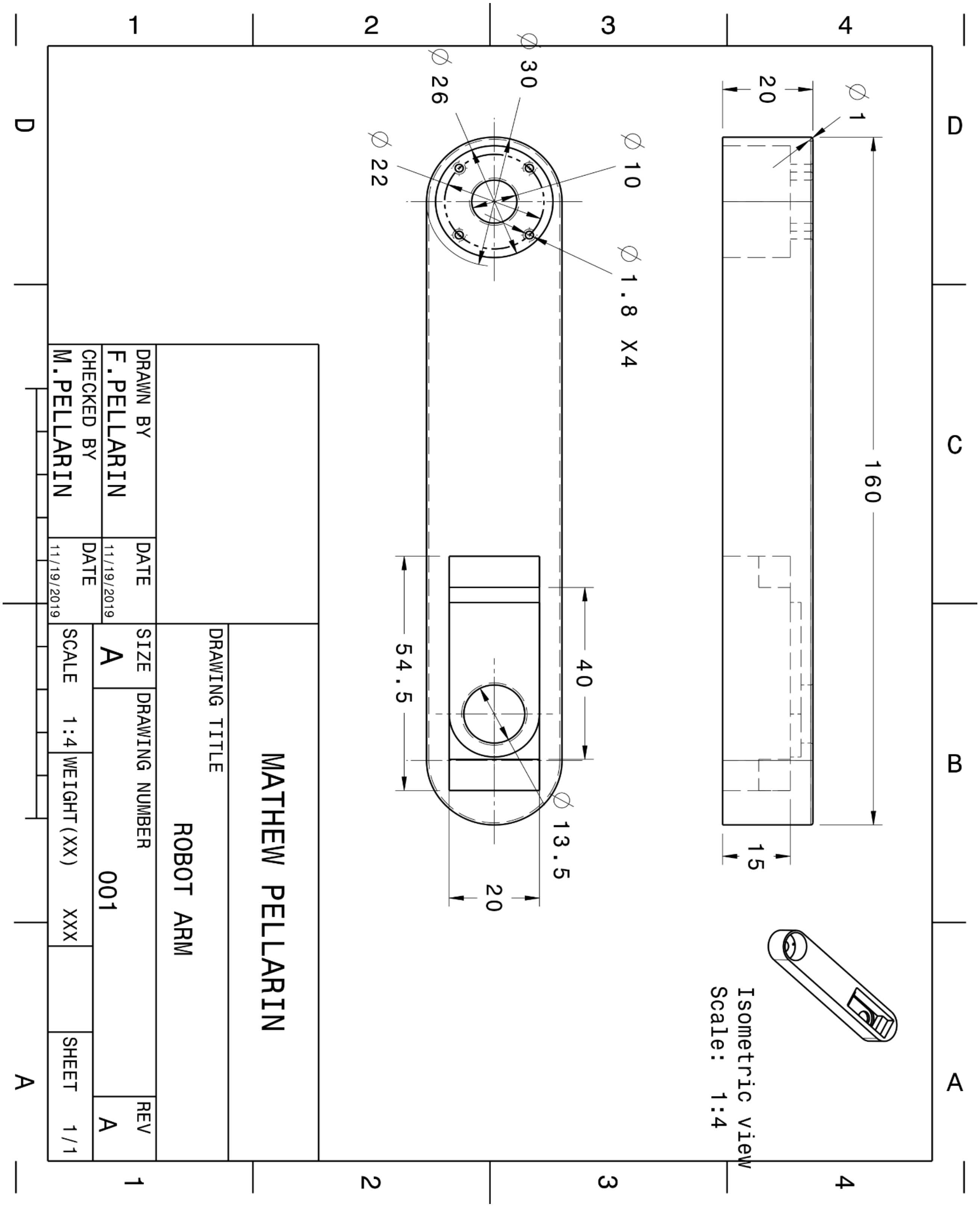
ST CLAIR COLLEGE		TITLE: ARTICULATING ROBOTIC ARM	DATE: MAR'20	SHT:
DRN BY: M.M.P	CLASS: 001		SCALE: FULL	11

1-12

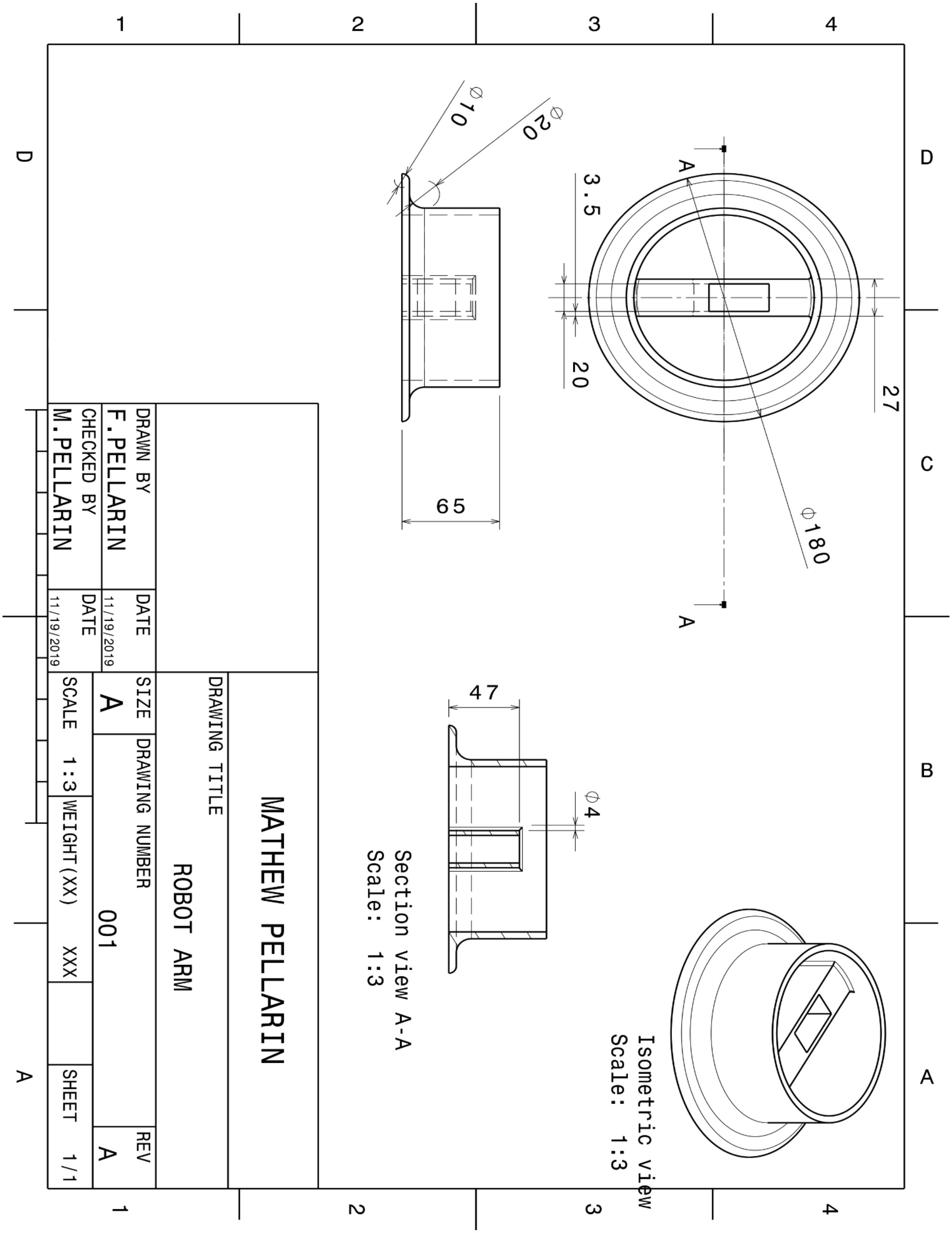

PRODUCED BY AN AUTODESK STUDENT VERSION
ST CLAIR COLLEGE
DRN BY:
M.M.P
CLASS: 001
TITLE:
ARTICULATING ROBOTIC ARM
DATE:
MAR'20
SCALE:
FULL
SHT:
12
12 01
12 02
12 03
12 04
12 05
12 06
12 07
12 08
12 09
12 10
12 11
12 12
12 13
12 14
1 09
END
010501 18AWG BLUE
090301 18AWG YEL
12 02
12 03
12 04
VCC
0VDC
1
2
3
4
580Ω LED
680Ω LED
PIC817
10kΩ
PROXIMITY SENSOR DRIVER BOARD 12V TO 5V
1202PRS
1203PRS
1204PRS
18AWG BLK
030601 18AWG YEL
3 06
12 07
12 08
12 09
0 VDC 18AWG BLK
2 14
END
CONV END PROXIMITY SENSOR
CONV MID PROXIMITY SENSOR
CONV START PROXIMITY SENSOR
CONV END PROXIMITY DRIVER
CONV MID PROXIMITY DRIVER
CONV START PROXIMITY DRIVER

Mechanical Drawings

2-1

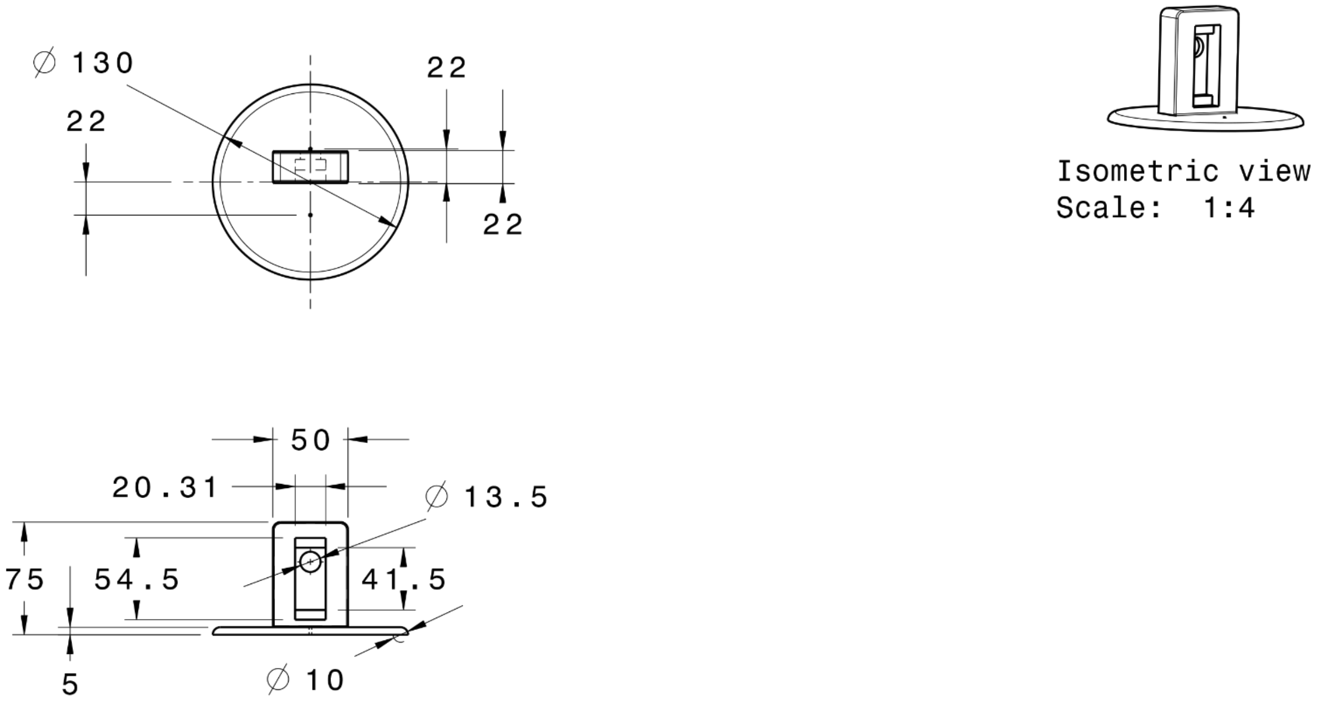


2-2



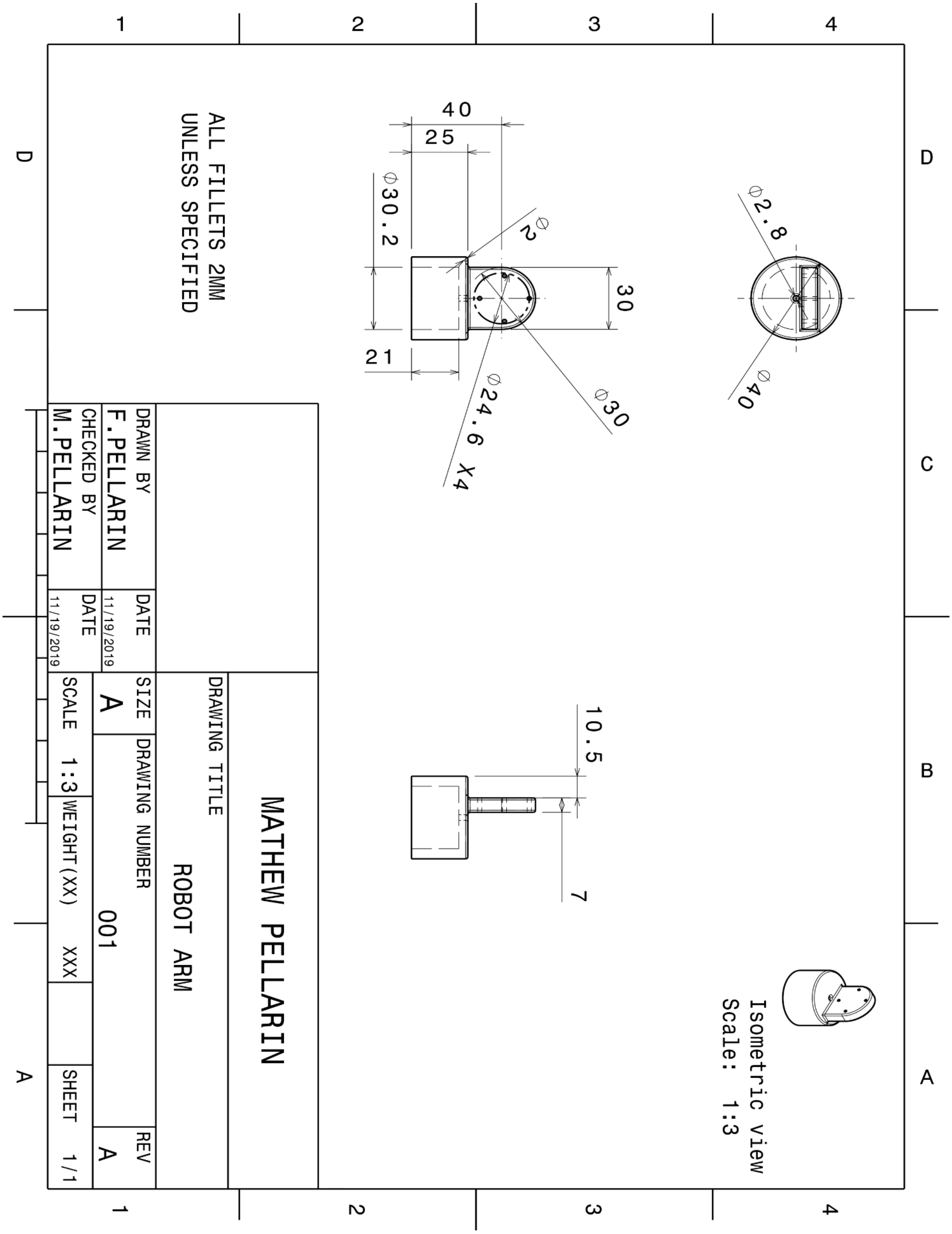
1
2
3
4
D
C
B
A
⌀10
⌀20
3.5
20
27
65
⌀180
A
A
47
⌀4
Section view A-A
Scale: 1:3
Isometric view
Scale: 1:3
MATHEW PELLARIN
DRAWING TITLE
ROBOT ARM
DRAWN BY
F.PELLARIN
DATE
11/19/2019
CHECKED BY
M.PELLARIN
DATE
11/19/2019
SIZE
A
DRAWING NUMBER
001
REV
A
SCALE 1:3
WEIGHT(XX) XXX
SHEET 1/1

2-3

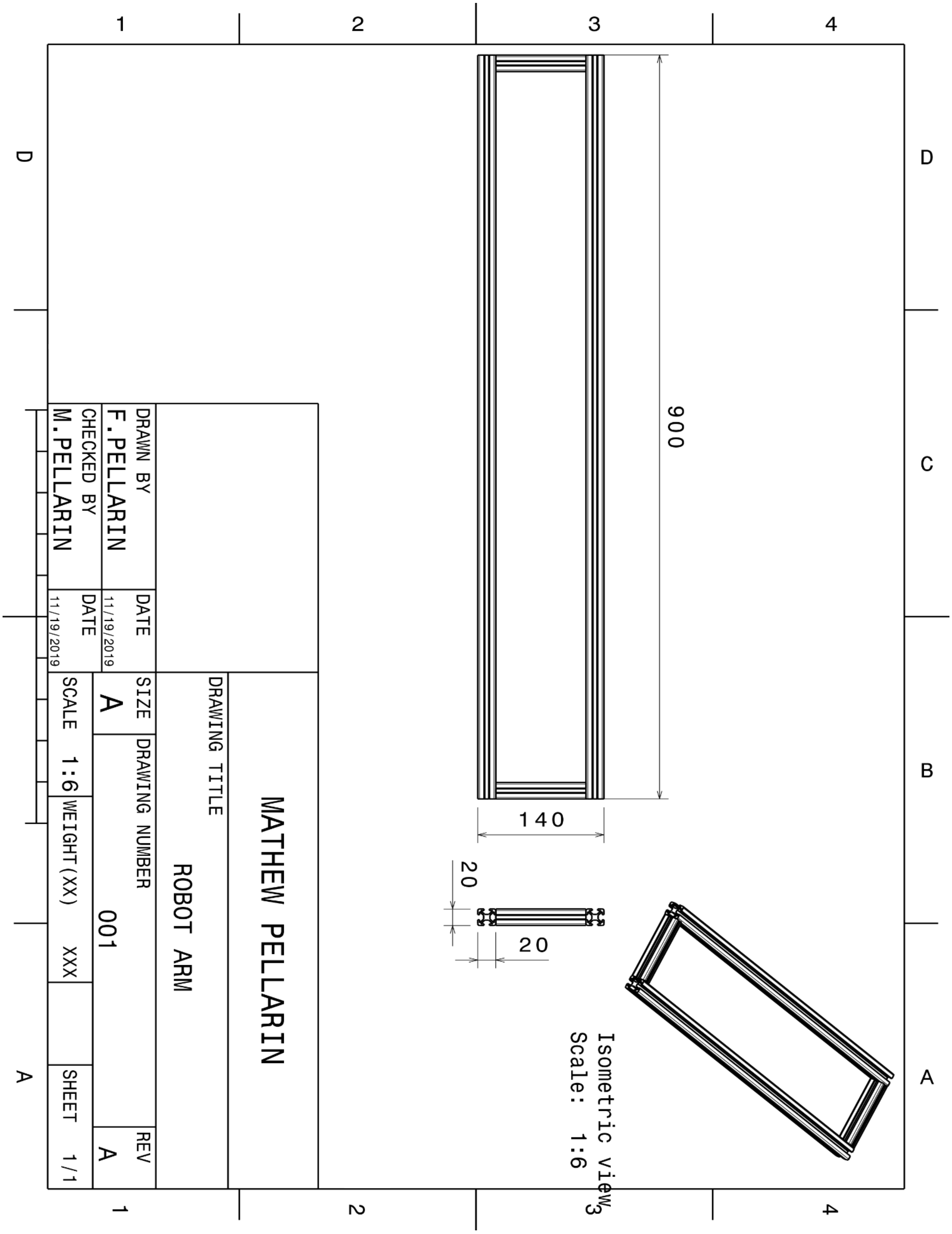


		MATHEW PELLARIN				
		DRAWING TITLE ROBOT ARM				
DRAWN BY F.PELLARIN	DATE 11/19/2019	SIZE A	DRAWING NUMBER 001			REV A
CHECKED BY M.PELLARIN	DATE 11/19/2019	SCALE 1:4	WEIGHT(XX) XXX		SHEET 1/1	

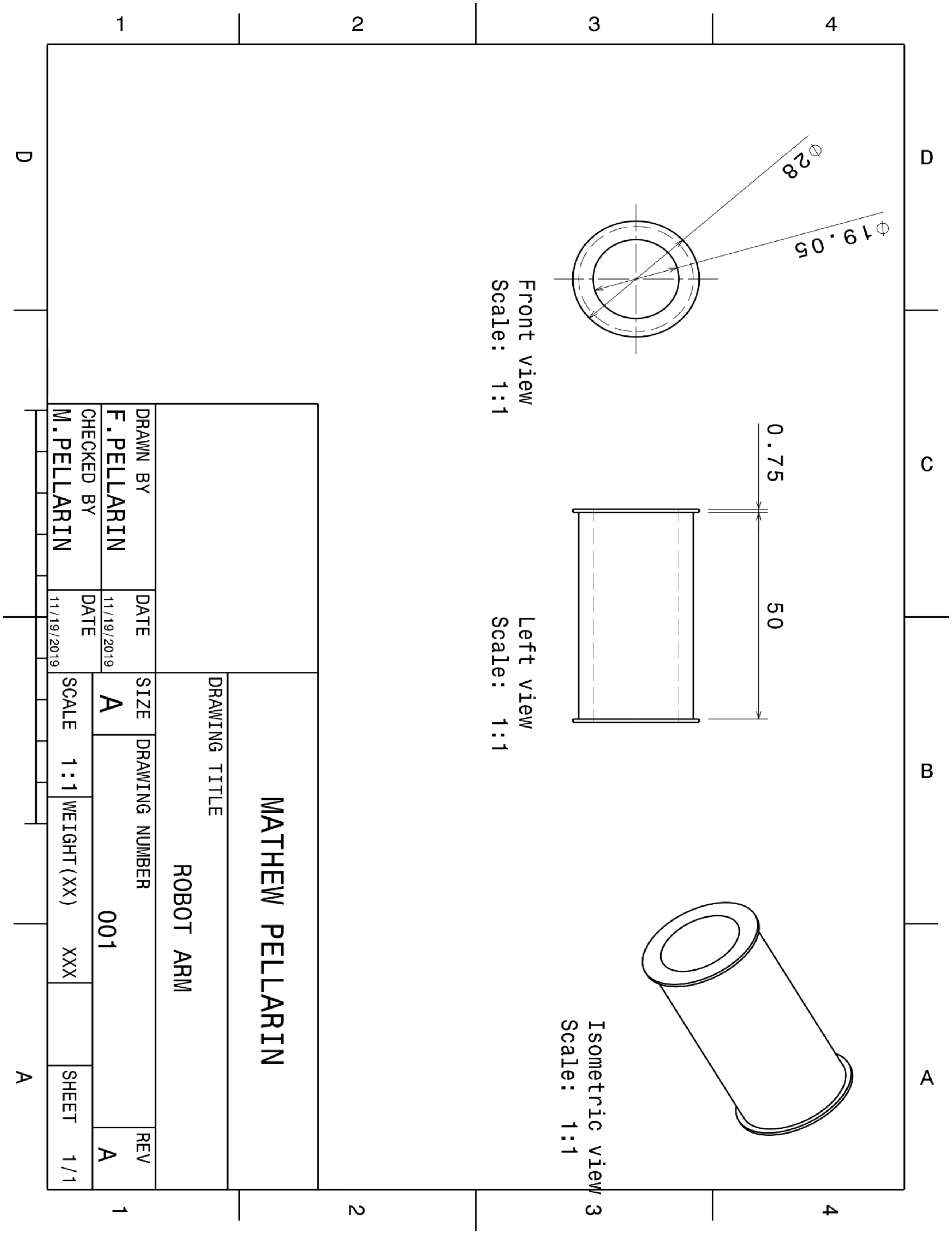
2-4


ALL FILLETS 2MM
UNLESS SPECIFIED
40
25
⌀30.2
⌀2
30
21
⌀24.6 X4
⌀30
⌀2.8
⌀40
10.5
7
Isometric view
Scale: 1:3
MATHEW PELLARIN
DRAWING TITLE
ROBOT ARM
DRAWN BY
F.PELLARIN
DATE
11/19/2019
CHECKED BY
M.PELLARIN
DATE
11/19/2019
SIZE
A
DRAWING NUMBER
001
REV
A
SCALE 1:3
WEIGHT(XX) XXX
SHEET 1/1
1
2
3
4
D
C
B
A

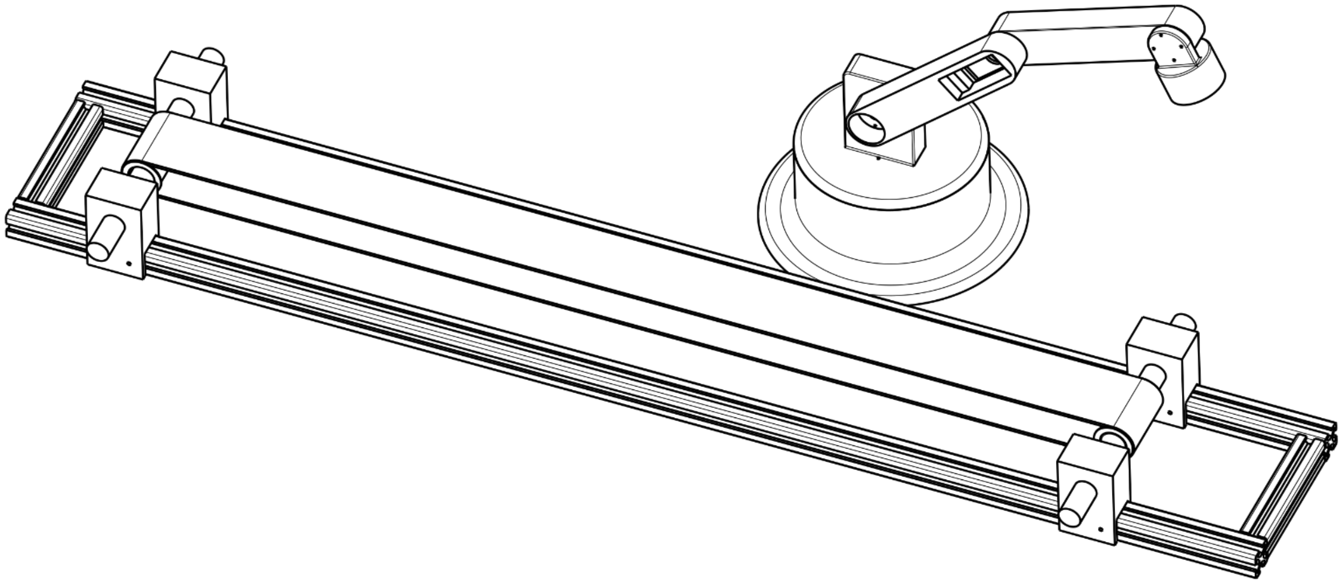
2-5


900
140
20
20
Isometric view
Scale: 1:6
MATHEW PELLARIN
DRAWING TITLE
ROBOT ARM
DRAWN BY
F.PELLARIN
DATE
11/19/2019
CHECKED BY
M.PELLARIN
DATE
11/19/2019
SIZE
A
DRAWING NUMBER
001
REV
A
SCALE
1:6
WEIGHT(XX)
XXX
SHEET
1/1
1
2
3
4
A
B
C
D

2-6


Front view
Scale: 1:1
⌀28
⌀19.05
Left view
Scale: 1:1
0.75
50
Isometric view
Scale: 1:1
DRAWN BY
F.PELLARIN
DATE
11/19/2019
CHECKED BY
M.PELLARIN
DATE
11/19/2019
MATHEW PELLARIN
DRAWING TITLE
ROBOT ARM
SIZE
A
DRAWING NUMBER
001
REV
A
SCALE
1:1
WEIGHT(XX)
XXX
SHEET
1/1
1
2
3
4
A
B
C
D

2-7



Isometric view
Scale: 1:4

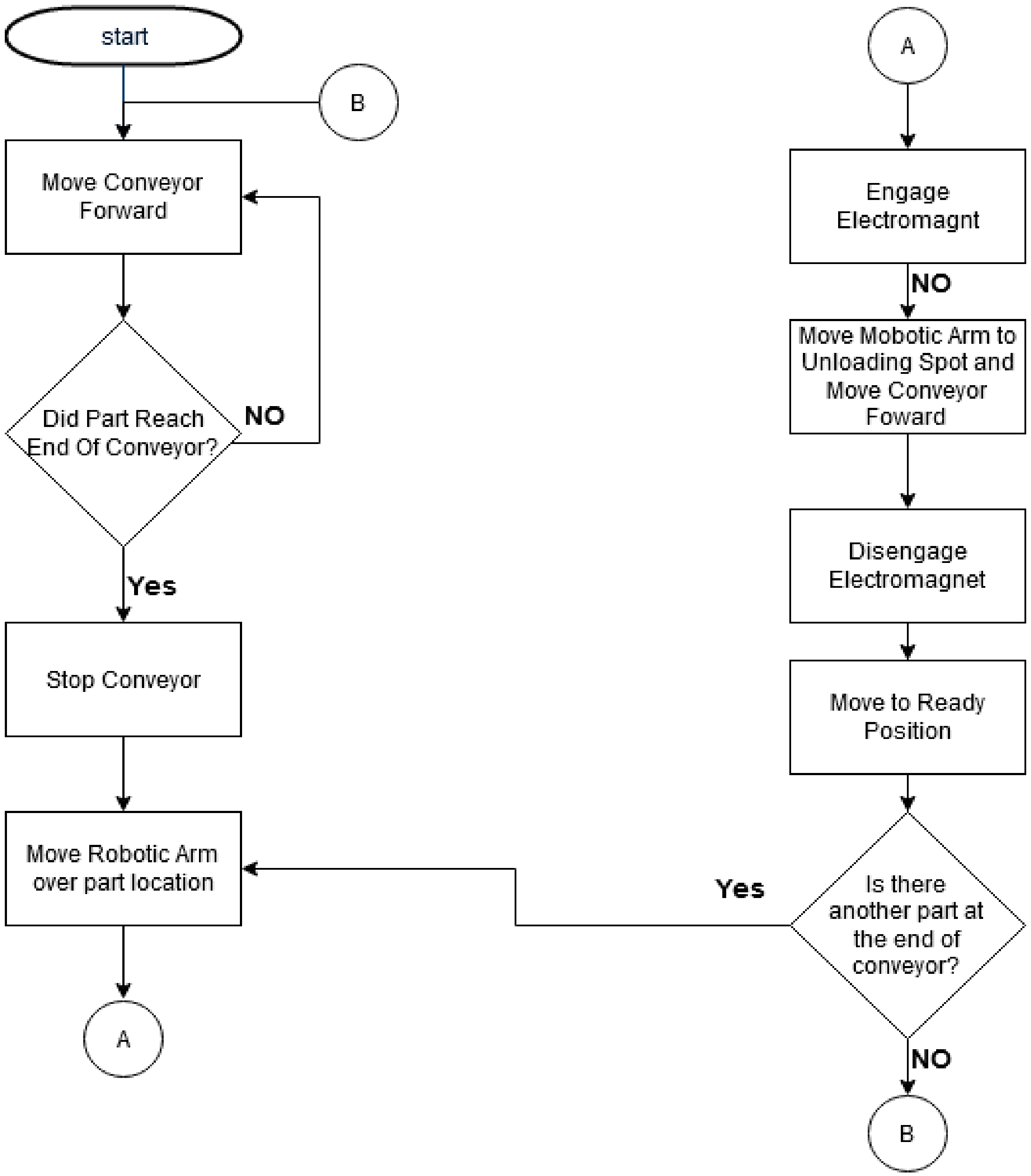
		MATHEW PELLARIN					
		DRAWING TITLE ROBOT ARM					
DRAWN BY F.PELLARIN	DATE 11/19/2019	SIZE A	DRAWING NUMBER 001				REV A
CHECKED BY M.PELLARIN	DATE 11/19/2019	SCALE	1:4	WEIGHT(XX)	XXX		SHEET 1/1

Flowchart

3-1

Articulating Robotic Arm

By: Mathew Pellarin


start
B
Move Conveyor Forward
Did Part Reach End Of Conveyor?
NO
Yes
Stop Conveyor
Move Robotic Arm over part location
A
A
Engage Electromagnt
NO
Move Mobotic Arm to Unloading Spot and Move Conveyor Foward
Disengage Electromagnet
Move to Ready Position
Is there another part at the end of conveyor?
Yes
NO
B

Arduino Code

4-1

```
#include <VarSpeedServo.h>

//Push Buttons
#define DecreasePBNO 53
#define IncreasePBNO 51
#define ServoSelectPBNO 49
#define ConvFWDPBNO 45
#define ConvREVPBNO 47
#define StopPBNO 43
#define StartPBNO 41
#define ManPBNO 39
#define AutoPBNO 37

//LEDs
#define RunningLED 23
#define ManLED 25
#define AutoLED 29
#define StopLED 35
#define Servo1LED 40
#define Servo2LED 33
#define Servo3LED 31
#define Servo4LED 38

//Stepper Motor
#define ConvEn 46
#define ConvDir 48
#define ConvPulse 5

#define Electromagnet 52

#define BaseServo 30
#define LowerServo 32
#define UpperServo 34
#define HeadServo 36

#define ConvProxStartNO 22
#define ConvProxMidNO 24
#define ConvProxEndNO 26

#define ConvyorON LOW
#define ConvyorOFF HIGH
#define ConvyorREV LOW
#define ConvyorFWD HIGH
#define ConvyorPWMON 127
#define ConvyorPWMOFF 0
```

4-2

```
VarSpeedServo BaseRCServo;
VarSpeedServo LowerRCServo;
VarSpeedServo UpperRCServo;
VarSpeedServo HeadRCServo;

int IncreasingServo = 0;
int DecreasingServo = 0;
int ServoSelectorVal = 1;
int ServoSelectorButtonState = LOW;
int ServoSelectorButtonLastState = LOW;
int Servo1State = 90;
int Servo2State = 90;
int Servo3State = 90;
int Servo4State = 90;
int PartOnConv = 0;

int ManualPBState = LOW;
int AutoPBState = LOW;
int StopPBState = LOW;
int StartPBState = LOW;
int RunningState = LOW;
int ManMode = LOW;
int AutoMode = LOW;
int ConvProxStartLastState = LOW;

int servostep = 2;

int BaseServoisMoving = LOW;
int LowerServoisMoving = LOW;
int UpperServoisMoving = LOW;
int HeadServoisMoving = LOW;
int ServoDone = LOW;

int endproxlaststate = LOW;
int holdingpart = LOW;
int stoptorunning = HIGH;
int Servo3Max = 0;
int i = 1;
int switchedtoauto = LOW;
int armready = LOW;
int stoplaststate = LOW;
int electromagnetlaststate = HIGH;
int Insequence = 0;

unsigned long cyclestarttime = 0;
```

4-3

```
int erroron = 0;
int erroroff = 0;
int blinkamount = 0;
int cleanconveyor = 0;

unsigned long ProxDebouncetime = 0;
unsigned long DebounceDelay = 25;

int finishscan = LOW;

const int PartLocationLength = 200;
intPartLocation[PartLocationLength];


void setup() {

  //Onboard Power LED
  pinMode(12, OUTPUT);
  digitalWrite(12, HIGH);

  //Initializing All Inputs
  pinMode(IncreasePBNO, INPUT);
  pinMode(DecreasePBNO, INPUT);
  pinMode(ServoSelectPBNO, INPUT);
  pinMode(ConvFWDPBNO, INPUT);
  pinMode(ConvREVPBNO, INPUT);
  pinMode(ManPBNO, INPUT);
  pinMode(AutoPBNO, INPUT);
  pinMode(StartPBNO, INPUT);
  pinMode(StopPBNO, INPUT);
  pinMode(ConvProxStartNO, INPUT);
  pinMode(ConvProxMidNO, INPUT);
  pinMode(ConvProxEndNO, INPUT);

  pinMode(ConvEn, OUTPUT);
  pinMode(ConvDir, OUTPUT);
  pinMode(ConvPulse, OUTPUT);
  pinMode(RunningLED, OUTPUT);
  pinMode(ManLED, OUTPUT);
  pinMode(AutoLED, OUTPUT);
  pinMode(StopLED, OUTPUT);
  pinMode(Servo1LED, OUTPUT);
  pinMode(Servo2LED, OUTPUT);
  pinMode(Servo3LED, OUTPUT);
  pinMode(Servo4LED, OUTPUT);
  pinMode(Electromagnet, OUTPUT);
```

4-4

```
  //Set Default Outputs
  digitalWrite(ConvEn, LOW);
  digitalWrite(Electromagnet, HIGH);
  digitalWrite(RunningLED, LOW);
  digitalWrite(ManLED, LOW);
  digitalWrite(AutoLED, LOW);
  digitalWrite(StopLED, HIGH);

  Serial.begin(9600);
}

void loop() {
  //Cycle time Couter and Debounce
  cyclestarttime = millis();

  //Read button inputs
  ManualPBState = digitalRead(ManPBNO);
  AutoPBState = digitalRead(AutoPBNO);
  StopPBState = digitalRead(StopPBNO);
  StartPBState = digitalRead(StartPBNO);

  //Start Machine if START PB is pressed
  if(StartPBState == HIGH && StopPBState == LOW){
    digitalWrite(StopLED, LOW);
    RunningState = HIGH;
    digitalWrite(RunningLED, HIGH);

    //if it was in OFF state, Reset everything
    if(stoptorunning == HIGH){
      BaseRCServo.attach(BaseServo);
      LowerRCServo.attach(LowerServo);
      UpperRCServo.attach(UpperServo);
      HeadRCServo.attach(HeadServo);
      Servo1State = 100;
      Servo2State = 110;
      Servo3State = 180;
      Servo4State = 0;
      BaseRCServo.write(Servo1State);
      LowerRCServo.write(Servo2State);
      UpperRCServo.write(Servo3State);
      HeadRCServo.write(Servo4State);
      stoptorunning = LOW;
    }
  }
```

4-5

```
  //Turn off Machine if STOP PB pressed
  if(StopPBState == HIGH){
    digitalWrite(RunningLED, LOW);
    digitalWrite(AutoLED, LOW);
    digitalWrite(ManLED, LOW);
    digitalWrite(Servo1LED, LOW);
    digitalWrite(Servo2LED, LOW);
    digitalWrite(Servo3LED, LOW);
    digitalWrite(Servo4LED, LOW);
    digitalWrite(StopLED, HIGH);
    AutoMode = LOW;
    ManMode = LOW;
    RunningState = LOW;
    stoptorunning = HIGH;
    ConvyorControl(ConvyorOFF, ConvyorFWD, ConvyorPWMOFF);
    digitalWrite(Electromagnet, HIGH);
    BaseRCServo.stop();
    LowerRCServo.stop();
    UpperRCServo.stop();
    HeadRCServo.stop();
  }

  //If Machine is running
  if(RunningState == HIGH){

    //Change Machine to Manual Mode
    if(ManualPBState == HIGH && AutoPBState == LOW){
      digitalWrite(AutoLED, LOW);
      AutoMode = LOW;
      digitalWrite(ManLED, HIGH);
      ManMode = HIGH;
    }

    //Change Machine to Auto Mode
    else if(AutoPBState == HIGH && ManualPBState == LOW &&
digitalRead(ConvProxEndNO) == LOW && AutoMode == LOW){
      digitalWrite(ManLED, LOW);
      ManMode = LOW;
      digitalWrite(Servo1LED, LOW);
      digitalWrite(Servo2LED, LOW);
      digitalWrite(Servo3LED, LOW);
      digitalWrite(Servo4LED, LOW);
      digitalWrite(AutoLED, HIGH);
      AutoMode = HIGH;
      armready = HIGH;
      switchedtoauto = HIGH;
```

4-6

```
      Insequence = 0;
    }

    //If Conveyor is not Clear and Auto is pressed
    else if(AutoPBState == HIGH && ManualPBState == LOW &&
digitalRead(ConvProxEndNO) == HIGH && AutoMode == LOW){
          cleanconveyor = 1;
    }

    //Change machine to No mode selected if both buttons pressed
    else if(AutoPBState == HIGH && ManualPBState == HIGH){
    digitalWrite(AutoLED, LOW);
    digitalWrite(ManLED, LOW);
    digitalWrite(Servo1LED, LOW);
    digitalWrite(Servo2LED, LOW);
    digitalWrite(Servo3LED, LOW);
    digitalWrite(Servo4LED, LOW);
    AutoMode = LOW;
    ManMode = LOW;
   ConvyorControl(ConvyorOFF, ConvyorFWD, ConvyorPWMOFF);
    }

    //Run Manual Mode
    if (ManMode == HIGH){
      manualmode();
    }

    //Run Auto Mode
    else if (AutoMode == HIGH){
      automode();
    }

    //Run conveyor Not Clean Blink Lights
    if (cleanconveyor == 1){
      cleanconv();
    }
  }

  //Delay needed to make smooth operation
  delay(20);
  finishscan = LOW;

  //Uncomment to Allow Debuging show Serial Monitor
  /*
   * debugging();
   * Serial.print(millis() - cyclestarttime);
```

4-7

```
  */
}

void cleanconv(){
  //How long lights should toggle for
  int timedelay = 12;

  //Turn Lights ON For specified time
  if(erroroff == 0 && erroron != 30){
   digitalWrite(Servo1LED, HIGH);
   digitalWrite(Servo2LED, HIGH);
   digitalWrite(Servo3LED, HIGH);
   digitalWrite(Servo4LED, HIGH);
   digitalWrite(AutoLED, HIGH);
    erroron++;
  }

  //Turn Lights OFF For specified time
  if(erroron == timedelay){
   digitalWrite(Servo1LED, LOW);
   digitalWrite(Servo2LED, LOW);
   digitalWrite(Servo3LED, LOW);
   digitalWrite(Servo4LED, LOW);
   digitalWrite(AutoLED, LOW);
    erroroff++;
  }

  if(erroroff == timedelay){
    erroron = 0;
    erroroff = 0;
    blinkamount++;
  }
  if(blinkamount == 2){
    cleanconveyor = 0;
    blinkamount = 0;
  }
}

void automode(){
  //If there is a part at end, stop conveyor
  if(digitalRead(ConvProxEndNO) == HIGH){
   ConvyorControl(ConvyorOFF, ConvyorFWD, ConvyorPWMOFF);
  }
  else{
   ConvyorControl(ConvyorON, ConvyorFWD, ConvyorPWMON);
  }
```

4-8

```
  //Check to See if any Servos are still in Motion
  BaseServoisMoving = BaseRCServo.isMoving();
  LowerServoisMoving = LowerRCServo.isMoving();
  UpperServoisMoving = UpperRCServo.isMoving();
  HeadServoisMoving = HeadRCServo.isMoving();
  if(BaseServoisMoving == HIGH || LowerServoisMoving == HIGH || UpperServoisMoving
== HIGH || HeadServoisMoving == HIGH){
    ServoDone = LOW;
  }else{
    ServoDone = HIGH;
  }

  //If Part has reached end of Conveyor start sequence of picking part up
  if(digitalRead(ConvProxEndNO) == HIGH && endproxlaststate == LOW){
    if(ProxDebouncetime == 0 || (millis() - ProxDebouncetime) > DebounceDelay){
      Insequence++;
    }
  }
  endproxlaststate = digitalRead(ConvProxEndNO);

  //If Servos had made it to New Position, Go to Next Position
  if(ServoDone == HIGH){
    if(Insequence >= 1){
      ArmPosEnd(i);
      //Turn on Electromagnet when on First Sequence
      if(i == 1){
        digitalWrite(Electromagnet, LOW);
        holdingpart = HIGH;
      }
      if(i == 2){
        ProxDebouncetime = millis();
      }
      if(i == 3){
        ProxDebouncetime = 0;
      }
      //Turn off Electromagent when on Last Sequence
      else if(i == 5){
        digitalWrite(Electromagnet, HIGH);
        holdingpart = LOW;
        Insequence--;
        i = 0;


        //if(digitalRead(ConvProxEndNO) == HIGH){
        //  Insequence++;
```

4-9

```
        //}
      }
      i++;
    }
  }
}

void manualmode() {
  //Read Button State
  IncreasingServo = digitalRead(IncreasePBNO);
  DecreasingServo = digitalRead(DecreasePBNO);
  ServoSelectorButtonState = digitalRead(ServoSelectPBNO);

  //Change to Next Servo
  if(ServoSelectorButtonState == HIGH && ServoSelectorButtonLastState == LOW &&
IncreasingServo == LOW && DecreasingServo == LOW){
    ServoSelectorVal++;
    if(ServoSelectorVal == 5){
      ServoSelectorVal = 1;
    }
  }
  ServoSelectorButtonLastState = ServoSelectorButtonState;
  //Update Servo Selected LED
  ServoLEDSelect();

  //Servo Motions
  if (IncreasingServo == HIGH || DecreasingServo == HIGH && ServoSelectorButtonState
== LOW){
      Servomotion();
  }

  //Conveyor Belt
  ConvControl();
}

void ServoLEDSelect(){
  //Display the Selected Servo
  switch(ServoSelectorVal){
    case 1:
     digitalWrite(Servo1LED, HIGH);
     digitalWrite(Servo2LED, LOW);
     digitalWrite(Servo3LED, LOW);
     digitalWrite(Servo4LED, LOW);
    break;

    case 2:
```

4-10

```
      digitalWrite(Servo1LED, LOW);
      digitalWrite(Servo2LED, HIGH);
      digitalWrite(Servo3LED, LOW);
      digitalWrite(Servo4LED, LOW);
    break;

    case 3:
      digitalWrite(Servo1LED, LOW);
      digitalWrite(Servo2LED, LOW);
      digitalWrite(Servo3LED, HIGH);
      digitalWrite(Servo4LED, LOW);
    break;

    case 4:
      digitalWrite(Servo1LED, LOW);
      digitalWrite(Servo2LED, LOW);
      digitalWrite(Servo3LED, LOW);
      digitalWrite(Servo4LED, HIGH);
    break;
  }
}

void Servomotion() {
  //Determine Max of Servo 3 so arm cannot hit conveyor
  Servo3Max = Servo2State * 2 + 48;

  //Determine which servo is selected and increase od decrease
  switch(ServoSelectorVal){
    case 1:
      if (IncreasingServo == HIGH && DecreasingServo == LOW){
        if (Servo1State < 180){
          Servo1State = Servo1State + servostep;
        }
      }
      else if (DecreasingServo == HIGH && IncreasingServo == LOW){
        if (Servo1State > 0){
          Servo1State = Servo1State - servostep;
        }
      }
      break;

    //reversed due to reverse Servo
    case 2:
      if (IncreasingServo == LOW && DecreasingServo == HIGH){
        if (Servo2State < 150){
          Servo2State = Servo2State + servostep;
```

4-11

```
    }
  }
  else if (DecreasingServo == LOW && IncreasingServo == HIGH){
    if (Servo2State > 40){
      if(Servo2State < 60 && Servo1State <= 146){
        if(Servo3Max > Servo3State){
          Servo2State = Servo2State - servostep;
        }
      }
      else{
        Servo2State = Servo2State - servostep;
      }
    }
  }
  break;

case 3:
  if (IncreasingServo == HIGH && DecreasingServo == LOW){
    if (Servo3State < 180){
      if (Servo2State > 60){
        Servo3State = Servo3State + servostep;
      }
      else{
        if(Servo3State < Servo3Max){
          Servo3State = Servo3State + servostep;
        }
      }
    }
  }
  else if (DecreasingServo == HIGH && IncreasingServo == LOW){
    if (Servo3State > 0){
      Servo3State = Servo3State - servostep;
    }
  }
  break;

//reversed due to reverse Servo
case 4:
  if (IncreasingServo == LOW && DecreasingServo == HIGH){
    if (Servo4State < 180){
      Servo4State = Servo4State + servostep;
    }
  }
  else if (DecreasingServo == LOW && IncreasingServo == HIGH){
    if (Servo4State > 0){
      Servo4State = Servo4State - servostep;
```

4-12

```
        }
      }
      break;
  }

  //Write New Servo Value
 BaseRCServo.write(Servo1State);
 LowerRCServo.write(Servo2State);
 UpperRCServo.write(Servo3State);
 HeadRCServo.write(Servo4State);
}

void ConvControl(){
  if (digitalRead(ConvFWDPBNO) == HIGH && digitalRead(ConvREVPBNO) == LOW){
   ConvyorControl(ConvyorON, ConvyorFWD, ConvyorPWMON);
  }
  else if (digitalRead(ConvREVPBNO) == HIGH && digitalRead(ConvFWDPBNO) == LOW){
   ConvyorControl(ConvyorON, ConvyorREV, ConvyorPWMON);
  }
  else{
   ConvyorControl(ConvyorOFF, ConvyorFWD, ConvyorPWMOFF);
  }
}


void ConvyorControl(int ConveyorEn, int ConveyorDir, int ConveyorPulse){
   digitalWrite(ConvEn, ConveyorEn);
   digitalWrite(ConvDir, ConveyorDir);
   analogWrite(ConvPulse, ConveyorPulse);
}


void ArmPosEnd(int pos){
  int SS1 = 0;
  int SS2 = 0;
  int SS3 = 0;
  int SS4 = 0;

//end conveyor pos
  switch(pos){
    case 1:
      SS1 = 70;
      SS2 = 46;
      SS3 = 144;
      SS4 = 16;
    break;
```

4-13

```
//home offset pos
    case 2:
      SS1 = 70;
      SS2 = 110;
      SS3 = 180;
      SS4 = 0;
    break;

//over drop off pos
    case 3:
      SS1 = 180;
      SS2 = 110;
      SS3 = 180;
      SS4 = 0;
    break;

//drop off pos
    case 4:
      SS1 = 180;
      SS2 = 42;
      SS3 = 166;
      SS4 = 48;
    break;

//Home pos
    case 5:
      SS1 = 100;
      SS2 = 110;
      SS3 = 180;
      SS4 = 0;
    break;
  }

  //write new servo values
  BaseRCServo.write(SS1, 40, false);
  LowerRCServo.write(SS2, 30, false);
  UpperRCServo.write(SS3, 30, false);
  HeadRCServo.write(SS4, 100, false);
}

void debugging(){
  //if debugging is on, display all values
  Serial.println(" ");

  if(ManMode == HIGH){
```

4-14

```
    Serial.print("IncPB = ");
   Serial.print(digitalRead(IncreasePBNO));
    Serial.print("\tDecPB = ");
   Serial.print(digitalRead(DecreasePBNO));
    Serial.print("\tSSPB = ");
   Serial.print(digitalRead(ServoSelectPBNO));
    Serial.print("\tConvFWD= ");
   Serial.print(digitalRead(ConvFWDPBNO));
    Serial.print("\tConvREV = ");
   Serial.print(digitalRead(ConvREVPBNO));
    Serial.print("\tAutoPB = ");
   Serial.print(digitalRead(AutoPBNO));
    Serial.print("\tManPB = ");
   Serial.print(digitalRead(ManPBNO));
    Serial.print("\tStartPB = ");
   Serial.print(digitalRead(StartPBNO));
    Serial.print("\tStopPB = ");
   Serial.print(digitalRead(StopPBNO));

    Serial.print("\tSSVal = ");
   Serial.print(ServoSelectorVal);
    Serial.print("\tServo = ");
   Serial.print(Servo1State);
    Serial.print("  ");
   Serial.print(Servo2State);
    Serial.print("  ");
   Serial.print(Servo3State);
    Serial.print("  ");
   Serial.print(Servo4State);
    Serial.print(" S3Max ");
   Serial.print(Servo3Max);
  }

  Serial.println(" ");

 Serial.print(digitalRead(ConvProxEndNO));
 Serial.print(digitalRead(ConvProxMidNO));
 Serial.print(digitalRead(ConvProxStartNO));

  Serial.print("\tCycle Time\t");
}
```