REMOTE CONTROLLED ROBOT ARM

By

Mengen Lu

A capstone project submitted for
Graduation with University Honors

April 2020

University Honors
University of California, Riverside

APPROVED

_____
Dr. Konstantinos Karydis
Department of Electrical and Computer Engineering

_____
Dr. Richard Cardullo, Howard H Hays Jr. Chair, University Honors

Abstract

During the Senior Design, my group and I developed a remote controlled robot arm. The final prototype includes two major parts: the glove controller and the robot arm. My contributions to this project include establishing a customized communication protocol, designing power system, calibrating flex sensors and servos, and measuring current ratings. Many design preparations and analysis were performed at the beginning of the project. The considerations include the selection of many components, such as microcontroller, finger motion tracking devices, arm and shoulder motion tracking devices, and the communication devices. In addition, we devised the system and software architecture based on the selected components. Flex sensors, MPU-6050 gyroscopes, HC-05 Bluetooth, and Arduino Nano, were installed on the glove side. The robot arm has four SG90 Micro Servos for each finger and four MG995 Metal Gear Servos for wrist, elbow and shoulder joints. After the construction, my teammates and I designed specific experiments to test the functionality of the system. These tests include observing Bluetooth data transmission rate, decoding the received information from Bluetooth, and connecting the multi-meter to measure the current ratings through the servos.

Table of Contents

INTRODUCTION

The electrical modules on the glove included Arduino Nano, gyroscopes, flex sensors, a Bluetooth HC-05 module and LEDs. The microcontroller, Arduino Nano, sampled all the sensors on the glove. Flex sensors on the glove were designed to track the user's finger and elbow positions. The two gyroscopes on the user’s wrist and shoulder reflected joint rotation angles. The LEDs installed on the breadboard indicate the current status of the system during initialization. The button helped to transit from the current initialization phase to the subsequent ones. The Bluetooth module sends the data acquired by the Arduino Nano.

The actual robot arm is in a separate and different system. The Bluetooth HC-05 module in the robot system was designed to only receive data from gloves, which indicates the user's finger and arm positions. There are eight servos mounted on the robot arm: four SG90 Micro Servos and four MG995 Metal Gear Servos. After receiving the user’s movement data, the Arduino Mega converted that information to PWM signals. Also, it distributed the signals in terms of voltages to the servos. The SG90 Micro Servos were used to move the fingers. The MG995 Metal Gear Servos were used to move the robot's wrist, elbow, and shoulder.

This project required students think like engineers and implement theories learned from lectures to the real world. It gives every engineer opportunities to face challenging problems and to come up with solutions. The overall objective of this project was to give practical training and better prepare electrical engineers for the future through this experience.

PERSONAL CONTRIBUTIONS TO PROJECT

Bluetooth Communication and Software Development

I was responsible for coming up with solutions to achieve high speed wireless communication for our project. After a thorough research about the subject on the internet, I decided to use the Bluetooth module for this project due to its cheap price and plenty of online documentation about its setup. The success of our final project partially depends on whether Bluetooth could communicate and transfer data to each other. During the fall quarter 2019, I found out that there were two modes of configuration and each required different circuit setup. Even though I was able to set up Bluetooth to transfer data, the communication was buggy and unreliable and occasionally generated garbage. After reading many documentations for this bug in winter 2020, I designed a communication protocol, which ensured that microcontrollers could receive data from Bluetooth even if both microcontrollers worked asynchronously. I applied my communication protocol in software and programmed Arduino Nano to transfer data and Arduino Mega to receive data on Arduino IDE software. The majority part of Bluetooth function is on the slave file. Despite establishing communication protocol, I also wrote the majority part of the code for both the master and slave system. The code on the master file includes all functions for system initializations except for gyroscopes, getting and mapping user finger position, and sending data string. The entire program for the slave system was solely written by me.

System Power Design and Calculation

Another major contribution to this project is figuring out a way to properly power the entire system. I did a lot of research online regarding the servos' voltage and current because

they are the key components on the robotic arm. I talked to Professor Chomko and asked him questions about the servos' electrical behavior and characteristics. He explained to me that when the servos were not in operation, they were basically open circuits. Once the servos started operating, they could be considered as variable resistors, of which the resistance would decrease if heavier load is applied, thus drawing more current from the power source. I calculated the current of all components on our glove and estimated that the total current was less than 100mA. However, for the servos on the robot arm, I needed to connect an amp meter to the circuit and measure servos' working current. After several measurements, which was carried out at solarium, I learned that the maximum total current for four micro-servo is about 1.3 A and 1.8 A for each MG995 servo. Based on the information from the professor and my practical current measurements, which set the design criteria for the power of our system, I bought the USB Charger which could supply up to 5V/2.4A for each of six ports and had a total power of 60W. To harness the USB power, I bought six USB male breakout boards and soldered 5V and ground pins. This setup enables my team to power all servos without using power supply from the school lab.

Flex Sensor Calibration

My learning and calibrating flex sensor took place in the first few weeks during the Senior Design. Since I had a good knowledge of how potentiometers work, I measured the resistance of each flex sensor when it's flat as well as when it's bent, and quickly came up with the circuit layout for those sensors. My teammate Mohammad suggested me to implement a user friendly feature, which calculates flex sensor maximum and minimum position during user

initialization phase. I was able to transform his idea to software, so now the initialization phase is more user friendly.

Servo Current Measurements and Servo Angle Calibration

In order to measure servos' working currents, I had to program Arduino and write multiple test files to see the maximum current in each unique scenario, such as that the servos are receiving continuous position values from the microcontroller and that they are not. Despite measuring servo's current, I also took part in calibrating the servo's angle with my teammates. During the making of the robotic arm, my teammate needed to tie the string around each finger on the servos, so I had to program a microcontroller to reset servo to either 0 degree or 180 degrees.

DESIGN CONSIDERATIONS

At the beginning of the design process, many design parameters and variables were taken into consideration. Those considerations include but not limited to the selection of microcontroller, software, and motion tracking and communication hardware.

Selection of microcontroller for this project

One of the major concerns includes determining the type of software and microcontroller to be implemented in this project. The type of system hardware chosen affects the selection of software because the hardware usually comes with its own software or programming environment. The decision to which microcontroller to implement has to be made early on so that engineers have sufficient time to read and study the documentations of the chosen

microcontroller's internal structure as well as its software environment. There are a variety of available options for microcontrollers on the market, such as Arduino Uno, Arduino Mega, Arduino Nano, Raspberry Pi, and Teensy. Arduino and teensy boards can be programmed in Arduino IDE software, which uses C language. Raspberry Pi has a stronger on board processor and comes with many functionalities, which runs mainly on Python scripts in Linux environment. Raspberry Pi is also capable of running multiple tasks at the same time due to its multi thread processing functionality. However, even though Raspberry Pi is more powerful and surpasses Arduino in numerous aspects, it still has limitations when it comes to controlling servos and motors. While Raspberry Pi's built in processor is good at dealing with software applications, such as installing libraries and performing complex task algorithms, it lacks the ability to interface and control multiple sensors given the limited number of pins. In addition, Raspberry Pi does not have onboard storage, so any improper hardware handling or software shutdown can possibly corrupt the files [1]. Arduino, on the other hand, comes with onboard storage, so no additional storage unit like SD card is needed. A developer can also benefit from Arduino's plenty of digital and analog pins because the robotic projects often involve various wire connections to either digital or analog pins. The microcontrollers' selections are therefore narrowed down to Arduino.

There are three major types of Arduino Uno, Arduino Mega, and Arduino Nano on the market. The key differences between these three types are the board size and the number of onboard pins. Arduino Nano has the smallest size among the three. It has 22 digital I/O pins and 6 of those pins are PWM [2]. In addition, Arduino Nano has 8 analog I/O pins, which can be used to sample the analog signals. Slightly bigger than the size of Arduino Nano, Arduino Uno comes with 14 digital I/O pins and 6 analog input pins [3]. Lastly, Arduino Mega 2560 board has

the biggest size among the three and it also has the most number of digital and analog I/O pins. Not to mention three sets of transmission and receiving ports, Arduino Mega has a total of 54 digital pins and 16 analog pins [4]. Since most of the sensors are going to be located on the user's hand, forearm, and shoulder, a lightweight microcontroller should be put on the user's arm. Despite being small and light, a microcontroller needs to have sufficient analog pins to be used for sensor sampling. Therefore, Arduino Nano was chosen as the master for the user's arm thanks to its miniature size and abundant analog pins. For the robot arm, there is no choice better than Arduino Mega because most electrical modules, such as servos, require PWM signals.

Selection of finger motion tracking devices

Besides deciding the microcontroller model for this project, we also need to find a solution to track the user's arm and finger movement. There are a few options for tracking finger movements. One solution is to use three terminal potentiometers, which consists of a three pin variable resistor and a turning knob. The resistance can be varied by turning the knob on the potentiometer. The advantage of using those potentiometers is that the analog readings from those sensors are often accurate and do not fluctuate much. However, the drawback is that those potentiometers are bulky and they add a lot of weight to the user's hand. If they were selected to track the user's finger movement, additional modification needs to be made to measure finger flex angles. This additional mechanism includes adding extra joints between the user's fingers and the potentiometers. Any slight movement on the finger should also move the joints and those joints then rotate the knob on the potentiometer. Another solution for finger tracking is to implement flex sensors. The flex sensor looks like a long flat tape, which has two terminals on one of its ends. The working principle of the flex sensors is similar to that of the three terminal

potentiometers, but instead of turning the knob, one has to bend the flex sensor in order to change its resistance. The advantages of using flex sensors is that they can easily fit on a person's hand and they are way much lighter compared to three terminal potentiometers. As long as those flex sensors are securely strapped to the back of the hand, they can report any person's finger movements to the microcontroller. Thus, flex sensors are the preferred finger movement tracking modules. Since all sensors take quantitative measurements differently, the two same types of sensors report different values. Flex sensor calibration is needed in the future to ensure data accuracy.

Selection of wrist, arm and shoulder motion tracking devices

Tracking of the user's arm and shoulder can be done by gyroscopes. Gyroscope is a motion sensing device, which is able to detect any rotation in a three dimensional environment [5]. To keep it simple, it tracks X, Y, and Z axis rotation of the user's arm and shoulder. Gyroscopes also use I2C communication protocol, which only needs serial clock (SCL) and serial data (SDA) connections. This communication protocol simplifies the communication between gyroscopes and the microcontroller, since only two connections are required. The plan is to use multiple gyroscopes, which are going to be located at different parts of the arm. Each gyroscope will receive a unique 7-bit address when using I2C protocol in order to differentiate itself from others.

Selection of communication devices

Speaking of communication, there are two communication options for this project. One option is to use the wired communication supported by I2C interface, and the other option is to

go wireless. Even though both wired I2C and wireless communication speeds are sufficient for the project, the effective communication distance of wired I2C reaches only up to 1 meter, whereas the wireless communication goes far beyond this range. Since the objective of this project is to achieve remote controlling of the robot arm, a wireless communication module becomes the priority. Bluetooth was eventually chosen as the wireless communication module.

SYSTEM ARCHITECTURE

There are four key modules on the glove. Those components include an Arduino Nano, two gyroscopes, four flex sensors, and a Bluetooth module. All components are powered by an external power supply. Since the glove is a low power device, it is sufficient to use the external power supply to power the circuits. Gyroscopes, flex sensors are the major sensors for detecting the user's finger and arm movement, so they are sampled by Arduino Nano on the glove. Arduino Nano integrates all that information and transmits it using Bluetooth.

The brain of the robot system is Arduino Mega, which controls all servos using PWM signals. The Bluetooth module receives the data and provides feedback to Arduino Mega. Arduino Mega then takes the information and generates the PWM signals accordingly. Four SG90 Micro Servos are connected in parallel and share a common power port. Each MG995 servo has its own port because each demands high current. The Bluetooth module, however, is powered by Arduino Mega due to its low working current.

Figure 1 is the electrical system of the glove. The system includes five major components: power supply, Arduino Nano, Gyroscopes, Flex sensors and Bluetooth. The external power bank provides energy to the Arduino Nano, which interacts with the rest of the

modules. Gyroscopes and flex sensors track user motion and give feedback to the Arduino Nano. Arduino Nano then sends the information to Bluetooth.

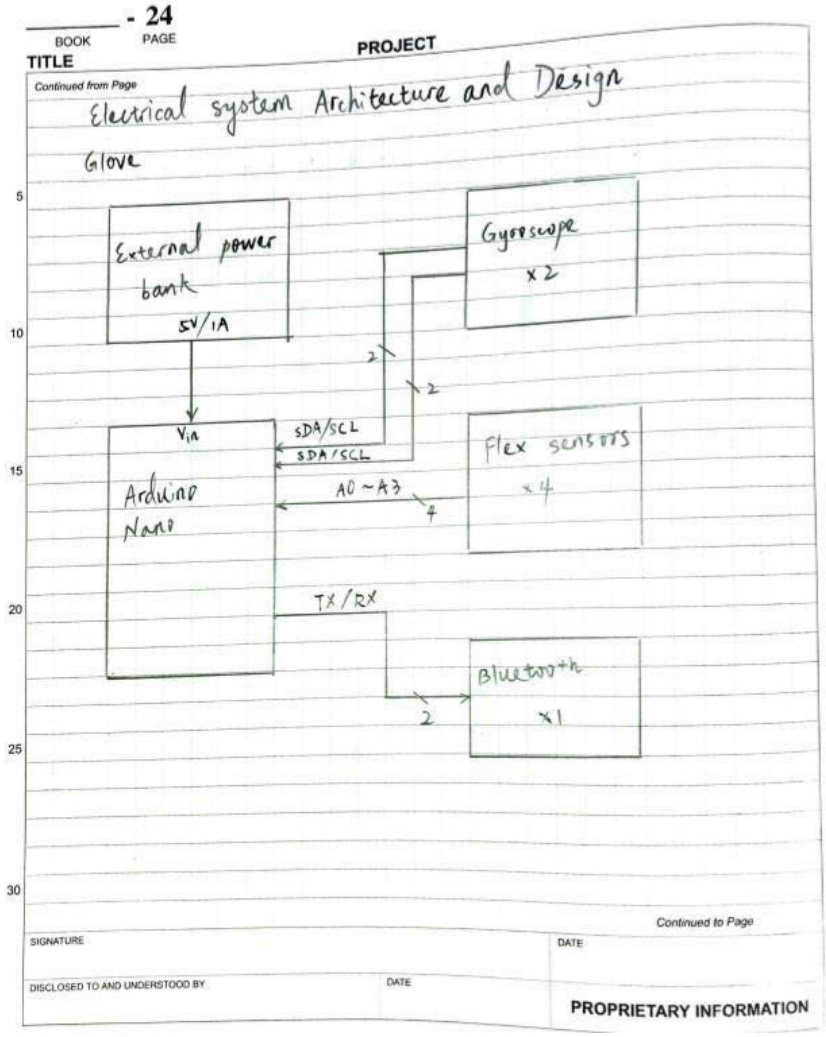


Figure 1. Electrical System Architecture and Design of Glove

Arduino Mega 2560 microcontroller controls the robot arm, since it has the most PWM signals. Arduino Mega outputs the PWM signal to control the angles of the servos. Since there are eight servos on the robot arm, eight PWM wires are connected to the servos. The power supply is capable of distributing at most 2.4 A of current to each servo. The SG90 Micro Servos

are low power devices, so power supply can easily handle four of them. Each MG995 Servo gets a 5 V port because the minimum current requirement for MG995 Servo is 2 A.

Figure 2 is the electrical system of the robot arm. This system includes USB Hub, Arduino Mega, Micro-Servos, MG995 Servos, and a Bluetooth. USB Hub supplies power to Arduino Mega, which interacts with all the servos and Bluetooth. Bluetooth provides position information to Arduino Mega, which then integrates that information and sends it to the servos.

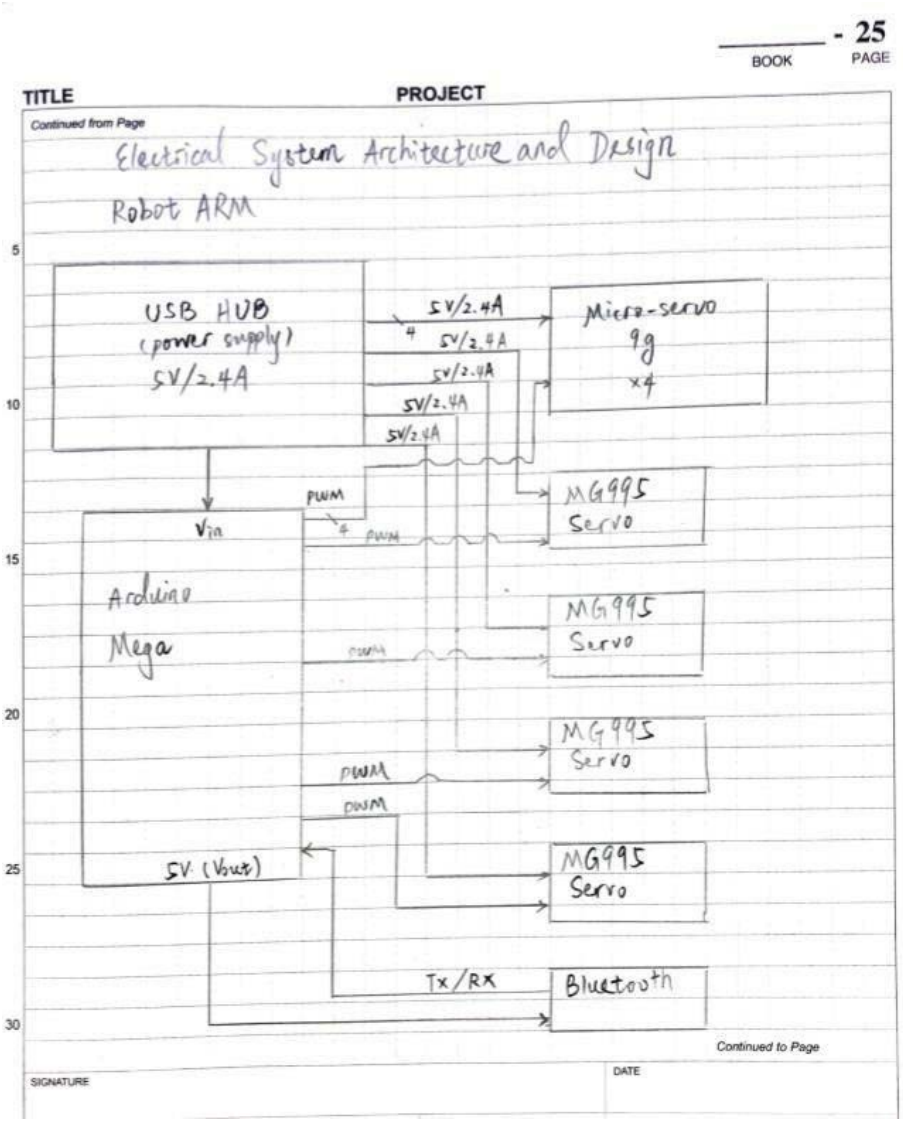


Figure 2. Electrical System Architecture and Design of Robot Arm

Figure 3 is a block diagram showing detailed connection of flex sensors, Bluetooth and Arduino Nano on the glove. The flex sensor are connected in series with the resistors and the analog pin measures the voltage across the flex sensor's two terminals. Bluetooth's VCC and GND are connected to 5 V power rail and the ground. Microcontroller's RX pin is connected directly to the Bluetooth's TX pin. Microcontroller's TX output is stepped down using a voltage divider and connects to RX pin on Bluetooth.

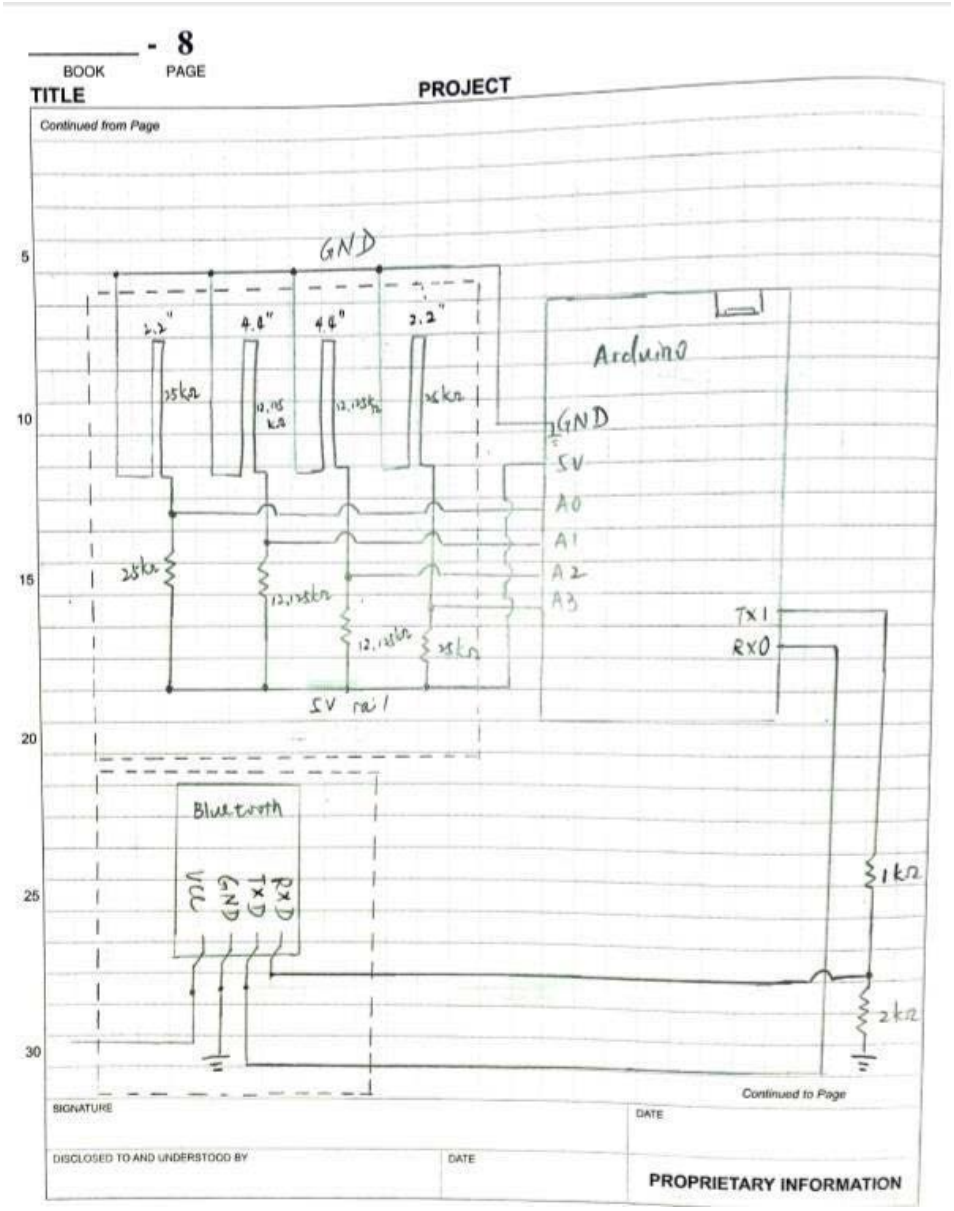


Figure 3. Electrical Connections on Glove

SOFTWARE ARCHITECTURE

Figure 4 block diagram shows how the software architecture of the glove runs and how the processes are queued and decided. The software on the glove system starts with an initialization phase, which involves calibrating LEDs, gyroscopes, and flex sensors. The LED initialization resets and turns off all the LEDs, which are used to indicate the progression during the initialization phase. Then the glove system calibrates the gyroscopes, which are used for tracking arm and shoulder motion. Gyroscope calibration takes about ten to fifteen seconds to finish. An LED turns on as soon as gyroscopes finish their initialization. Next the system samples the value from the flex sensor. Zero flex initialization samples the flex sensor values when the user straightens his or her fingers and arms. Full flex initialization samples the flex sensor values only when the user fully bends his or her fingers and arm. The initialization phase ends when the system finishes sampling flex sensor values. The code then enters an infinite loop, which continuously samples wrists, elbow, and shoulder movements. The gyroscopes take the values from the user's wrist, elbow, and shoulders, while flex sensors acquire data from the user's fingers. The sampled sensor data is then calculated and mapped into corresponding angles on each joint. Finally, the mapped angles for the joints are formatted and transmitted by Bluetooth. This loop continues until the system is powered off. The software for the glove system was developed by Mengen and Mohamad. Mohamad was responsible for creating the initialization and calibration phase. Mengen was responsible for value acquisition and using the Bluetooth module to communicate with the robotic arm to provide it with commands.

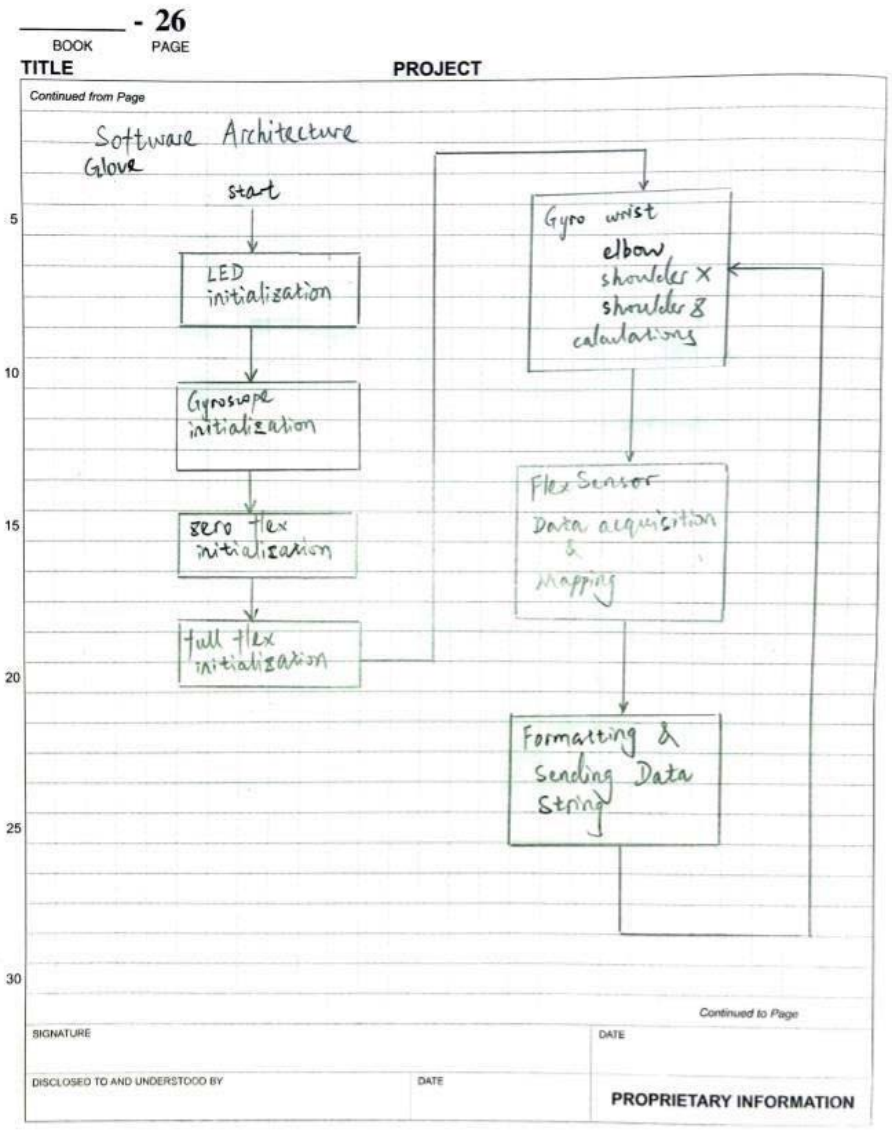


Figure 4. Software Architecture of Glove

Figure 5 block diagram shows how the software architecture of the robotic arm runs and how the processes are queued and decided. Similar to the software architecture on the glove, architecture on the robot arm also enters the setup phase first. There are two major components that require initialization: Bluetooth and Servo. Bluetooth initialization clears the buffer in the microcontroller and connects Slave Bluetooth to the Master Bluetooth. During Servo

initialization, the system creates Servo objects and assigns a digital output pin to each Servo object. Then the system enters a loop, which checks whether the incoming data is valid. If a valid string is found, then the system proceeds to extract information from the string. Otherwise, the system returns to receiving data. After extracting the needed angle information, the system distributes the angle information to specific variables. The Servo objects will then write those assigned variables to the output. The loop returns back to receiving data upon completion. The software for the robot arm was developed by Mengen.

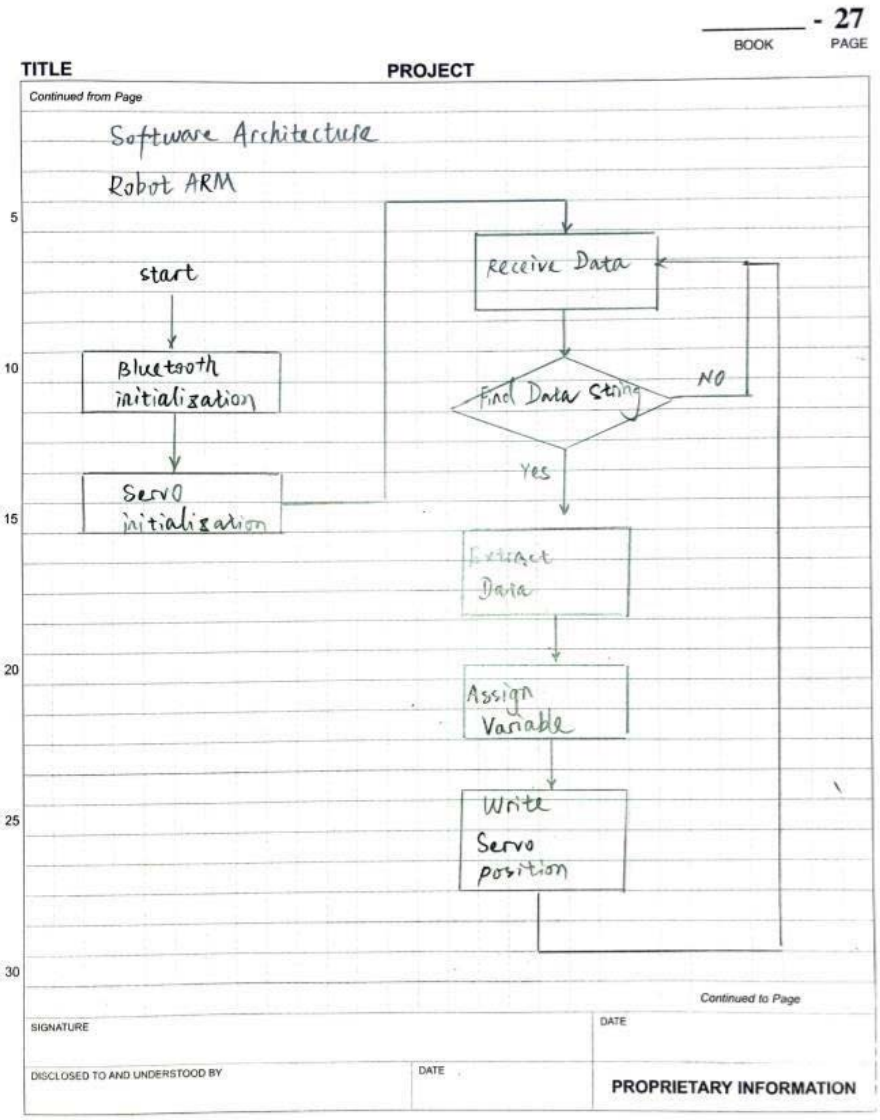


Figure 5. Software Architecture of Robot Arm

MODULES

Bluetooth

The default baud rate of Bluetooth HC-05 modules is 38400. Two Bluetooth modules were needed in order to set up the system. One Bluetooth functioned as a slave and the other as a master. In this design, the glove is the master and the robotic arm is the slave. Master Bluetooth converts the binary input to 5 V and 0 V, and sends this information through radio waves. The slave Bluetooth intercepts the radio wave and converts the radio wave to digital signal. Bluetooth has two modes of operation: AT mode and standard mode. AT mode is used to configure the Bluetooth modules, and the standard mode is used when the Bluetooth transmits or receives data. Holding down the button on the Bluetooth module allows programmers to enter AT mode. Slow flashing red LED is an indication that the module has entered its AT mode. Programmers can interact with Bluetooth by typing in commands on Serial Monitor in Arduino IDE software [6]. Make sure to configure the roles of Bluetooth and bind the address to establish secure connection. Bluetooth interfaces with Arduino microcontroller in this project. The communications are done on the TX and RX wires and the serial data is between 0 and 255, which corresponds to the decimal number on the ASCII table. Arduino microcontroller maps the data to the corresponding voltages, which is between 0 and 5 V. The mapped voltages are applied to the Bluetooth module through the TX wire. After sending the data, Bluetooth sends a signal back to the Arduino microcontroller through TX wire indicating task completion.

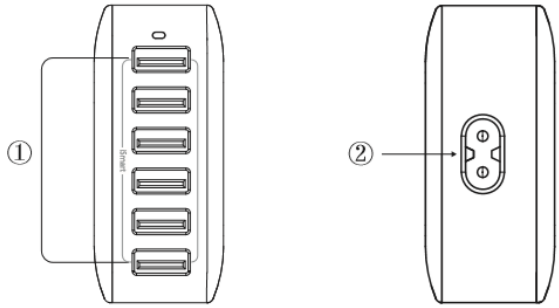
Power Supply

The power supply is a multiport USB charging station. It can supply 60 W and up to 12 A. There are a total of 6 ports on this device and each of the ports can tolerate up to 2.4 amps of

current. The 1.5 meter power cable is long enough to let the charger reach further or hide out of sight. RAVPower 6-Port Wall Charger is also fire-resistant and comes with a sleek matte black finish. The compact body is a new generation of the circuit design with overcharging, overheating, short circuit protection. This module can process 110 V to 240 V input. The 60 W power supply needs to be plugged to an outlet with a power cord. The outputs are female USB ports, so six 2.0 Type A male breakout boards were soldered to match the connections.

Figure 6.1 Power Supply Module

Source: Reference [7] (User Guide)



US UK CA

Product Diagram

① iSmart USB Output (DC 5V/2.4A Max Each)
② AC Cable Socket

DE

Produktdiagramm

① iSmart USB Ausgang (DC 5V/ max.2,4A jeweils)
② Netzkabelanschluss

Figure 6.2 Power Supply Side Views

Source: Reference [7] (User Guide)

Specifications

Model	RP-PC028
Input	AC 100-240V 50/60Hz 1.3A Max
Output	60W Total
iSmart USB Output	DC 5V/2.4A Max Each
Dimensions	10 x 7 x 2.6 cm / 3.9 x 2.7 x 1 in
Weight	192.8 g / 6.8 oz

Package Contents

- 1 x RAVPower 60W 6-Port USB Wall Charger (RP-PC028)
- 1 x Power Cord
- 1 x User Guide

Features

- Charges six devices simultaneously.
- iSmart charging technology automatically adapts to your devices and maximizes the charging speed.
- Multi-level protection circuitry ensures safe charging.

Caution

- Keep away from heat, flammables, poisons, and corrosive substances.
- Keep away from water, liquid, and moisture.
- Do not drop, knock, or disassemble the charger.
- Keep out of reach of children.

Figure 6.3 Description and Specification of Power Supply

Source: Reference [7] (User Guide)

EXPERIMENT DESIGN AND TESTING

Synchronizing Bluetooth

The objective was to determine whether an asynchronous communication was established in the system. Arduino Mega was set to receive data only if Arduino Mega finished all the other tasks. Arduino Nano was set to publish data regardless whether Arduino Mega received it. A serial port read function was placed inside a while loop code to ensure Arduino Mega received more than one complete string. The code then double checked whether a string with a size of less than 80 existed. The size limit was chosen to be 80 because this size was more than enough to store one string but not two. If this string was found, then the system would store this string. Otherwise, the receive function would keep running until the system found an acceptable string. A pair of Bluetooth modules needed to be set up to assess the feasibility of this system. The master Bluetooth's TX and RX pin should be connected to the RX and TX pin on Arduino Nano. The slave Bluetooth's TX and RX pin should be connected to the RX and TX pin on Arduino Mega. The results of the experiment could be seen from the serial monitor in Arduino IDE. The results are shown in the next section.

The performance of the communication protocol turned out to be exceptionally well during testing. The intrinsic delay of data transmission was less than 100 milliseconds. The time delay for Arduino Mega to check incoming data depended on how fast Arduino Mega processed the servos. For example, if it took Arduino Mega 1 second to finish processing the servo functions, then the total delay was about 1.1 seconds. The actual testing showed that there was a 200 millisecond delay from the servos. The total delay added up to 300 millisecond. Therefore, system refresh rate could reach up to 3 times per second.

Bluetooth Data Extraction Function Testing

The objective of this experiment was to check whether the code “Data Extraction” function could properly take apart the strings and distribute the data to variables. The test string was formatted in a specific way. The string started with the left arrow and ended with the right arrow. Each value was separated by a tab character. To test the function, a person could send a test string and print out the processed variable array on the serial monitor. The expected results should be exactly as the test string. For example, if the test string was “<111\t222\t333\t444\t555\t666\t777\t888\t999\t000>”, then the result should also be “<111\t222\t333\t444\t555\t666\t777\t888\t999\t000>”.

A customized test string “<111\t222\333\t444\t555\666\t777\t888\t999\765>” was sent to the slave Bluetooth during the actual testing. Then the data extraction function performed its duty and distributed the extracted data to a “tempArrayVal” variable. Figure 7.1 is a screenshot of the serial monitor, which demonstrated that the data was successfully extracted and assigned to “tempArrayVal” variables. The “tempArrayVal” was then used to print out values in a nicer format shown in the figure 7.2.

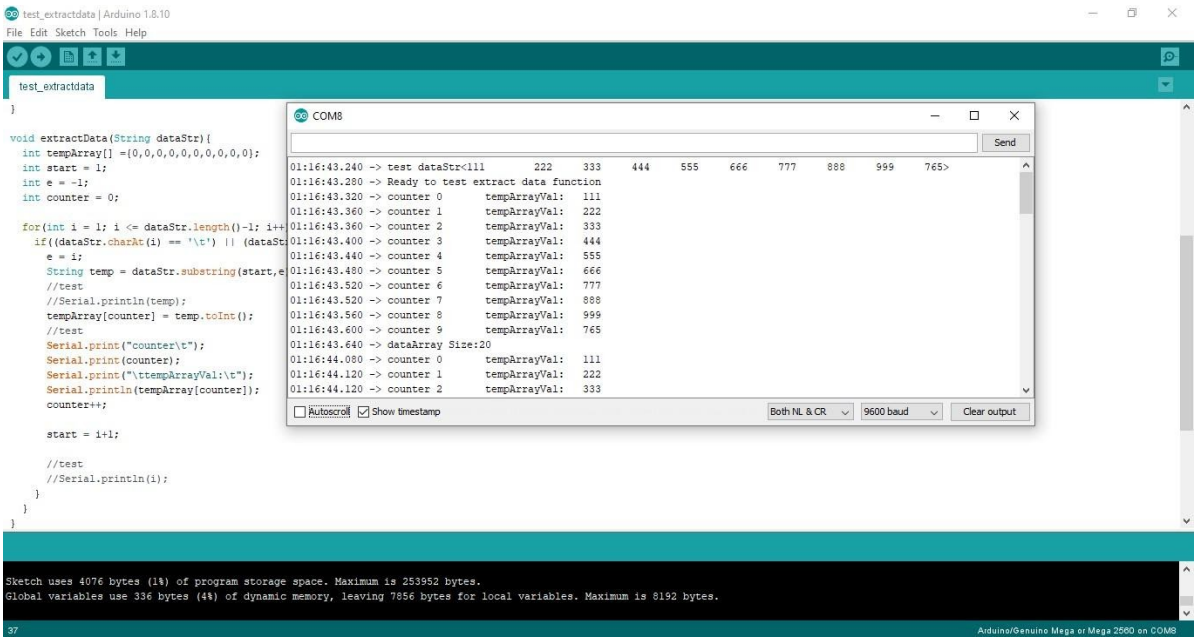


Figure 7.1 Extracted Data Printed Out On Serial Monitor

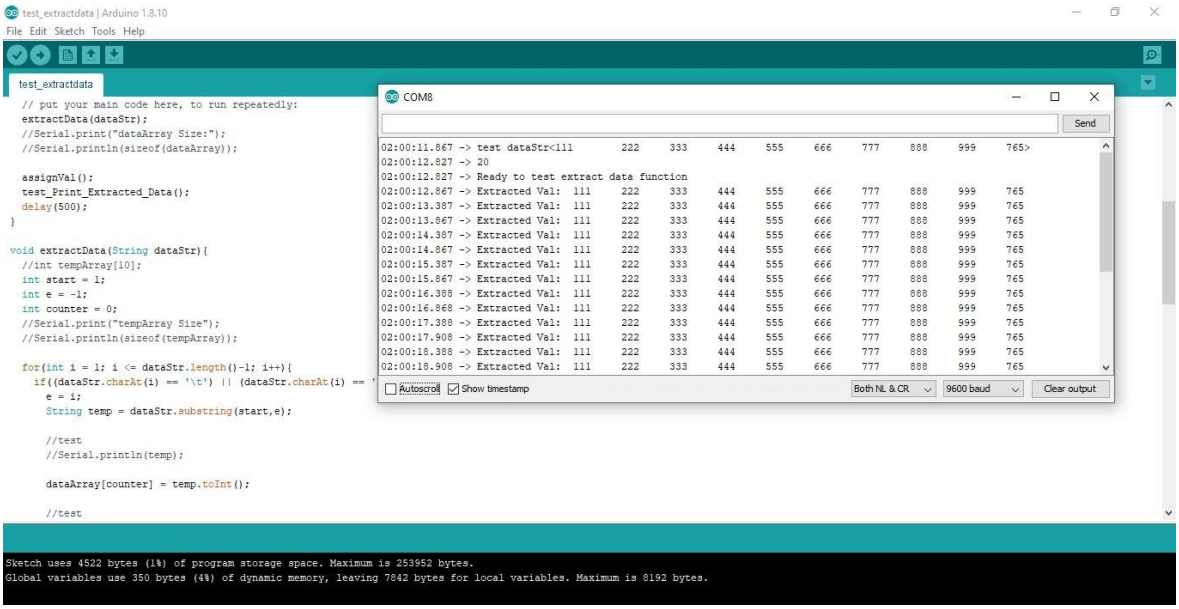


Figure 7.2 Formatted Data

Current Ratings Measurements

The objective was to discover the current ratings for the servos and glove system. An amp meter was connected in series with other modules in the circuit. The power supply from the lab

was used to power the circuits. After that, the tester could observe readings on a multimeter. The optimal current reading should be kept below 1 A for the glove. The combined current readings of four micro-servo should be less than 2.4 A. The current readings of each individual MG995 servo should also be kept below 2.4 A. In addition, the testing should observe the current readings when the servos' inputs were discrete.

The measured total current of all the modules excluding Arduino Nano was about 54 mA and 60 mA. Each port on the USB Hub could provide up to 5 V / 2.4 A. One port on the USB Hub thus would be used to power Arduino mega. One port would be used to power 4 micro servos. The rest four ports would be dedicated to each MG995 servo. Arduino mega would also be used to power Bluetooth because Bluetooth only consumes up to 35 mA during operation. In addition, some further testing revealed that the total current of four micro servos went up to 600 mA. Each MG995 servo consumed up to 1.8 A when being fully stalled.

CONCLUSION

The Bluetooth module and the communication protocol designed for this project allows a decent rate of data transmission on the system. The refresh rate is about two to three times per second for the robot arm. The power supply selected for the system is able to provide enough current for the servos to work properly. Since the power consumption on the glove is low, the circuit is powered directly from the microcontroller. The power system for the glove and the robot arm worked flawlessly and there was never an issue over the power supply in practical terms.

Through this project, I gained a deep understanding of electrical properties of various components, such as potentiometers, servos and power supply. I learned that servos are basically

variable resistors, whose resistance changes depending on the load. The lower the resistance, the higher the current that servos will draw from the circuit. In addition, I acquired knowledge of UART wireless communication, which uses TX and RX pins to transmit data. The data are transmitted in binary form and each character is about 10 bits. What's more, I am able to understand the power topic. The basic formula for power of the circuit is voltage multiple by the current. All electrical components have design requirements for voltage and current. Electrical engineering may be a challenging field, but you will find its beauty once inside this magical world. Electrical engineering is going to be my lifelong endeavor because I have a strong passion for this field.

ACKNOWLEDGEMENT

Given this is a team project, my teammates worked on other aspects of the project. Andres and Mohamad were responsible for the development of the electrical and the mechanical system of the project. They designed the structure of the glove and robot arm by using TinkerCad. They used modeling tools to 3D print the gears that would be moved by the servo motors attached to the frame. They also crafted the remote glove controller to incorporate a development breadboard. The Arduino Nano would tap onto the two MPU6050 units on the user's wrist and shoulder. The woodwork done on the frame was made with tools used from the Institute of Electrical and Electronics Engineers organization. All the woodwork and servo fittings were all performed by Andres, while George assisted him towards the end of this design project. Mohamad and George designed the stand, and then Andres and George assembled the stand to properly support all the gears and sensors. George, Mohamad and Mengen worked on testing components with the servos, gears, flex sensors, and other parts to ensure everything

operated accordingly. George, Mohamad, and Mengen tested current, resistance, and voltage ensuring all components are properly connected.

Special thanks to Professor Chomko for giving insights on servos' electrical property. Gustavo gave the group input on using a power bank or designing a PCB board to power the servos, and he told us that both would cost the same and gave personal experiences on how he used a PCB and power banks to power servos. Professor Karydis also gave this group ideas when brainstorming how to program this project and encouraged the use of arduinos. Also, Professor Karydis discussed with this group about the use of many of the components and his input was deeply appreciated in the process of this project.

REFERENCE

[1] "What are the differences between Raspberry Pi and Arduino?" 2017. [Online]. Available: https://www.electronicshub.org/raspberry-pi-vs-arduino/. [Accessed: 11-Mar-2020]

[2] "Arduino Nano TECH SPECS," 2020. [Online]. Available: https://store.arduino.cc/usa/arduino-nano. [Accessed: 11-Mar-2020]

[3] "Arduino Uno TECH SPECS," 2020. [Online]. Available: https://store.arduino.cc/usa/arduino-uno-rev3. [Accessed: 11-Mar-2020]

[4] "Arduino Mega TECH SPECS," 2020. [Online]. Available: https://store.arduino.cc/usa/mega-2560-r3. [Accessed: 11-Mar-2020]

[5] "MPU-6000 and MPU-6050 Product Specification Revision 3.4," 2013. [Online]. Available: https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/. [Accessed: 11-Mar-2020]

[6] "HC-03/05 Embedded Bluetooth Serial Communication Module AT command set," 2011. [Online]. Available: https://cdn.instructables.com/ORIG/F3O/K70G/H1LWQ0PO/F3OK70GH1LWQ0PO.pdf. [Accessed: 11-Mar-2020]

[7] “RAVPOWER 60W 6-PORT USB WALL CHARGER User Guide,” [Online]. Available: https://images-na.ssl-images-amazon.com/images/I/91+H6OS1AjL.pdf. [Accessed: 11-Mar-2020]