FRANKFURT UNIVERSITY OF APPLIED SCIENCES
Faculty 2: Computer Science and Engineering

VIETNAMESE-GERMAN UNIVERSITY (VGU)
Electrical & Computer Engineering (ECE)

Low-cost Manipulator using Manual Controller

by
Lưu Quang Minh
Matriculation No. 1273700

Supervisor
Dr.-Ing. Thai Truyen Dai Chan

Binh Duong new city, Viet Nam
27th April 2021

(This page intentionally left blank)

27th April 2021

Luu Quang Minh
ECE2017
Dinh Hoa dorm, Hoa Phu Street,
Binh Duong province, Viet Nam

Dr.-Ing, Thai Truyen Dai Chan, PhD
Senior Lecturer cum Academic Coordinator,
Electrical & Computer Engineering (ECE) study program
Vietnamese German University
Le Lai street, Hoa Phu ward
Thu Dau Mot city, Binh Duong, Vietnam

Dear Dr. Thai Truyen Dai Chan,

Enclosed is my Robotic Design Report, Low-cost Manipulator using Manual Controller, and is submitted to partially meet the requirements for the Senior project.

This report discusses the context for the project, gives a detailed description of the mechanical, electrical sub-systems and software design, addresses the feasibility of the project, and includes algorithm and all codes programmed for this project.

I appreciate the time you are taking to review this report and hope that it meets your approval. If you have any more questions, feel free to contact me either by telephone at 038.2424842 or e-mail: 13295@student.vgu.edu.vn.

Sincerely yours,

Minh

Minh, Quang Luu

Enclosure: “Low-Cost Manipulator using Manual Controller” (1 copy)

(This page intentionally left blank)

Disclaimer

I certify that the attached report is my original work. No other person's work has been used without due acknowledgment. Except where I have clearly stated that I have used some of this material elsewhere, it has not been presented by me for examination in any other course or subject at this or any other institution.

I understand that the work submitted may be reproduced and communicated to detect plagiarism. I understand that should this declaration be false, I am liable to be penalized under the Vietnamese-German University regulations.

Lưu Quang Minh
27th April 2021

Abstract

Low-cost Manipulator using Manual Controller
Luu Quang Minh

Nowadays, with the rapid development of technology, the important role of automatic machines is undeniable, which not only reduces the cost of labor but also operates with better effectiveness. One familiar topic of robotics is the robotic Arms. Therefore, this senior project will focus on building and coding a full-function 6 DOF Manipulator, while using a Web Server for Manual Controller.

For the project, its goal is to build a Robotic Arm with 6 DOF and proper functions that can be applied for different tasks and used for various purposes, including the idea of sorting fruits, helping student experiments, joining manufacturing processes. The Robotic Arm consists of three parts: the movement systems built with servos attached to different joints, signal transfer protocol via the Internet, and object-grabbing function. The Arm will be fixed on a stable shelf having the size of a standard wood chassis and is easily controlled from a built-in Web Server via Wifi in a LAN.

Key words: Manipulator, Robotic Arm, Servos, Web Server, Wifi, LAN

(This page intentionally left blank)

Acknowledgments

I would like to express my deep appreciation to Dr. Thai Truyen Dai Chan and Lab engineer Mr. Bien Minh Tri for their patient guidance, enthusiastic encouragement, and useful critiques of my project.
I would also like to thank all my friends and classmates for their suggestions and assistance during the writing of this proposal.
Finally, I wish to thank my parents for their support and encouragement throughout my study.

Luu Quang Minh
ECE2017

1.Table of Contents

2.Introduction

2.1. Objectives of the study

This Robotics Design Report for Low-cost Manipulator is submitted for partial fulfillment of the requirements of modules at the Vietnamese-German University and includes an overall description of the system, as well as descriptions of the mechanical, electrical, and software sub-systems.

2.2. Background

Senior Project is a very good opportunity for students to demonstrate the knowledge gaining so far during 6 semesters at VGU. Therefore, a Robotic Arm is a good choice to apply to this project because this upgraded prototype does not need any complicated process of mechanical assembles or code designs.

The project includes assembling frames from low-cost components, operating manipulators, and developing a Web Server by using some special boards. The project is also for learning new lessons about working with a network, which means building a local connection protocol for the prototype. Therefore, this upgraded prototype would help a lot in deeper understanding engineering design and telecommunication.

2.3. Scope and Limits of the report

This report is made for discussing the design, installation, and testing of the developed version from the prototype Robotic Arm using Arduino Uno R3 and NodeMCU DEVKIT 1.0. The Arm has to perform the task of lifting a heavy object. This report will not contain a working schedule or any other designs that are not needed for the project.

The Robotic Arm has some essential requirements that should be taken into account. Should be included first in the list is the connection protocol (TCP/IP) for remote control and here with LAN using a wireless network. This requirement is met by using static IP: 192.168.1.5 whereas the DHCP should be disabled before connecting. Then the system requires a friendly user interface, so a web page was created using HTML, javascript, and CSS. Besides, Bootstrap and font-awesome are included as two important tools for web design. Next is the requirement for six RC MG996R Tower pro Servos for smooth movement. The servo system should be strong enough to lift a 50-gram box as a prototype performance. Finally, together with a wireless controller, a physical controller using a USB cable (Serial Monitor) can be included in case no network access.

The following constraints was also identified that will limit the options available to meet the above-specified need. First, an ESP must be used for this project as the main communication microchip and be programmed by C++ programming language. Secondly, stable power is needed. Although this ESP8266 has some features which are compatible with each servo's electrical parameters (4.8-6V), six servos are signaling sensitive and they need a stable power supply adapter and enough current for signal transferring in order not to destroy the main Arduino board and also supply enough energy (actually not any power supply more than 5V can directly power the board). The servos work at 4.8 – 6VDC, 55g, and the electric traction is 9.4kg/cm (4.8V) or 11kg/cm (6V). Their working temperature must between 0^0C and 50^0C and their rated impulse withstand voltage is 4.8 – 7.2V. Moreover, the project should not include any actuators that can potentially hurt people or damage properties. Finally, there must be no plagiarism, as this heavily breaks the regulation and does no good for the study process.

3.Proposed systems

3.1. Description

This project involves the design, construction, and testing of an upgraded prototype Robotic Arm (RARMv2) controlled via the Internet. The required task for this manipulator is lifting a box or sorting fruit. The upper layer uses a NodeMCU DEVKIT1.0 for wireless communication, and the lower layer uses an Arduino Uno board for controlling servos. A web page HTML content used as User Interface (UI) has been created. By using TCP/IP, a communication protocol for easier transferring data between the Web page and ESP is created. Then the ESP built on NodeMCU DEVKIT and Arduino Uno can be connected via UART and therefore signals from the Web Server will be received and actions can be executed.

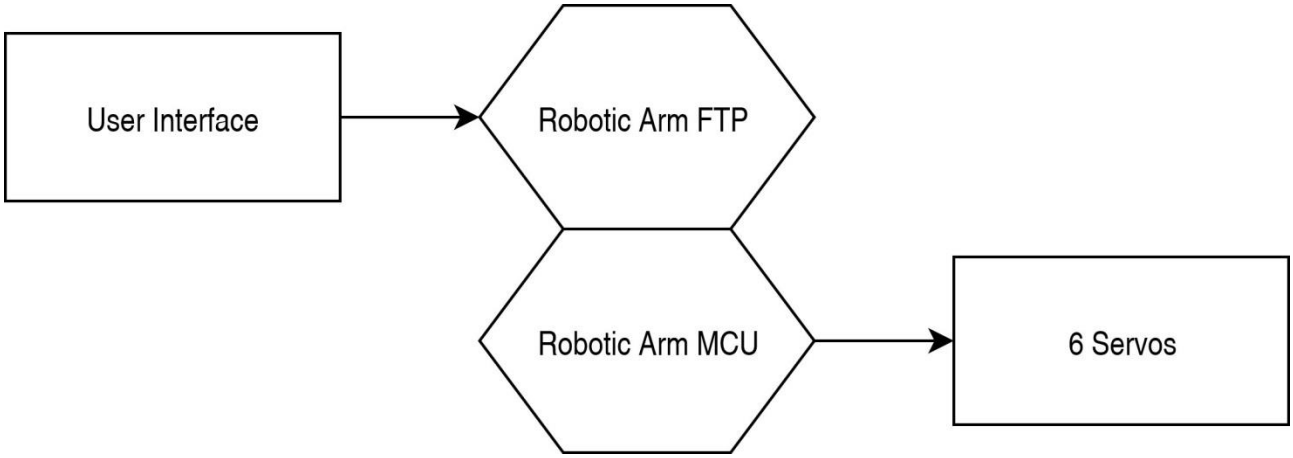


Figure 1: Context diagram for the Robotic Arm

3.2. Mechanical Sub-System

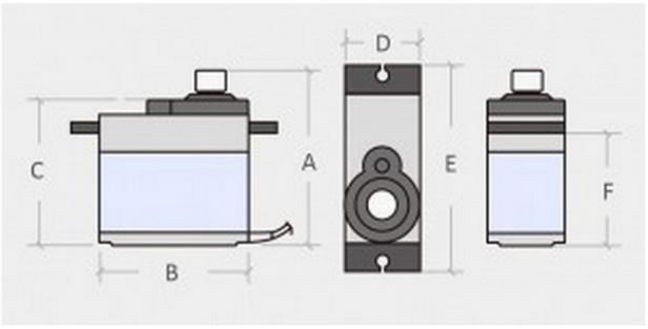
The Robotic Arm's hardware sub-system consists of Servos sub-system, Frame, and Power Supply. Servos are attached to the Frame which has six joints in total, while the Power Supply is the base of the whole system and it is also a counterweight since the calculated lifting capacity of the system is up to five kilograms.

3.2.1. Servo description

This Robotic Arm consists of 6 servos with 4 RC bronze-core servos MG996 and 2 RC Aluminum-core servos MG966 Tower pro (genuine). These servos are controlled by the pulses provided by the PWM pins of Arduino board. The MG966 Tower pro servo is the improved version of the MG965 Tower pro. Its PCB and IC control is upgraded to have better accuracy. The gear inside the servo is also boosted to improve dead bandwidth and centering. This servo is 40.7 cm high, 197 mm wide, and 429 mm long. The servo's gear is made of metal and it has 25 splines in total. The MG996 Tower pro weighs 55g, it has two voltage inputs: 4.8v and 6v. The turning speed depends on its two different voltage inputs: 0.20 second per 60 degrees for the input of 4.8 voltages and 0.16 second per 60 degrees for the input of 6 voltages. The gear type of this servo is digital and its working temperature is 0 to 55 Celsius degrees.[1]

Figure 2: Aluminum-core and Bronze-core Servo

After testing two types, It is observed that the Aluminum-core servo, a genuine servo, has a better protection circuit and larger electric power compared to the Bronze-core one. A brief calculation pointed out an economical solution: two genuine Servos and four Bronze-core Servos.



PRODUCT CONFIGURE TABLE

Weight(g)	55
Torque(kg)(4.8v)	9.4
Speed(sec/60deg)	0.17
A(mm)	42.7
B(mm)	40.9
C(mm)	37
D(mm)	20
E(mm)	54
F(mm)	26.8

Figure 3: Product configure table

3.2.2. Frame

The Frame is provided by H-Shop in separated parts. These parts include:

- Four U-shaped long brackets
- Three U-shaped brackets used for the base
- One mechanical gripper
- Six metallic servo wheels
- Four cup bearings
- Several screws and nuts

When being fully assembled, these parts make up a Grey Robotic Arm Frame with a picking system and 6 joints: the Base (Turning), the gripper (Gripper), the Wrist (Rotate and Joint1), the Elbow (Joint2), and the Shoulder (Joint3). The Arm is attached to a half-oval-shaped wooden piece that helps it standing still while being in action. The maximum distance the Arm can reach is up to 355 mm, the base is made to fit for carrying a servo and is spared enough space to let the wire go through. The final part of this Arm is the wooden piece, and this piece has a length of 215 mm, a width of 9 mm.[2]

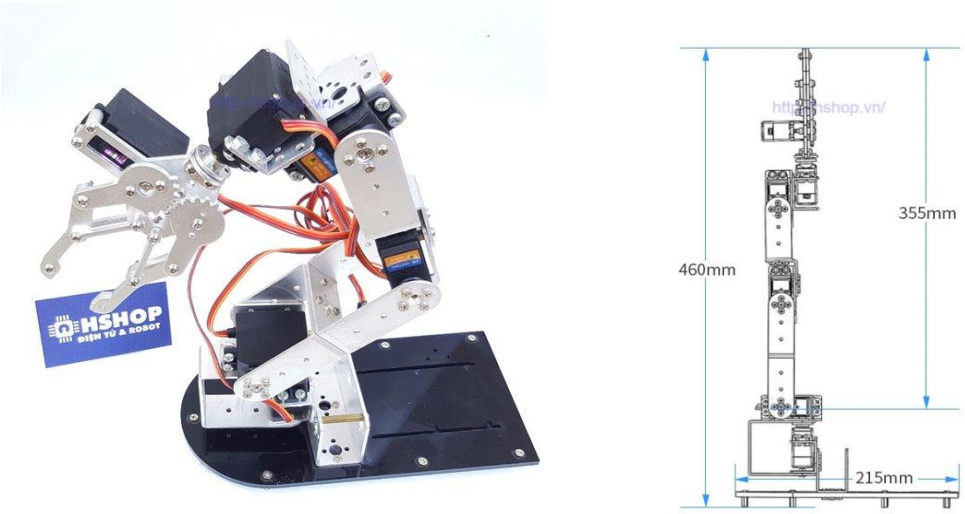


Figure 4: Robotic Arm Frame

3.2.3. Stable Power Supply

In this system, we used a 5V - 40A Power Supply exclusively for six servos. It is an enclosed Switching Power Supply. Its case is made from metal and its base is made from aluminum. It is about 200 mm in length, 100 mm in width, and 50 mm in height. Its input voltage is between 180V and 264V, at a frequency between 48 Hz and 63 Hz. Its output voltage is 5V, the maximum output power is 200W and output current is 40A. It operates at high temperatures (70^0 C) and has several protection systems including short-circuits, overload, overvoltage, over-temperature, and especially a radiator fan for reducing heat.[3]

Figure 5: AC 220V to DC 5V - 40A Switch Power Supply

3.3. Control Sub-System

The control sub-system consists of one Arduino Uno microcontroller board and one ESP8266 WIFI board, which are used for controlling the system and implementing communication protocol. These two boards need to be attached in a convenient way for easier wiring and good appearance.

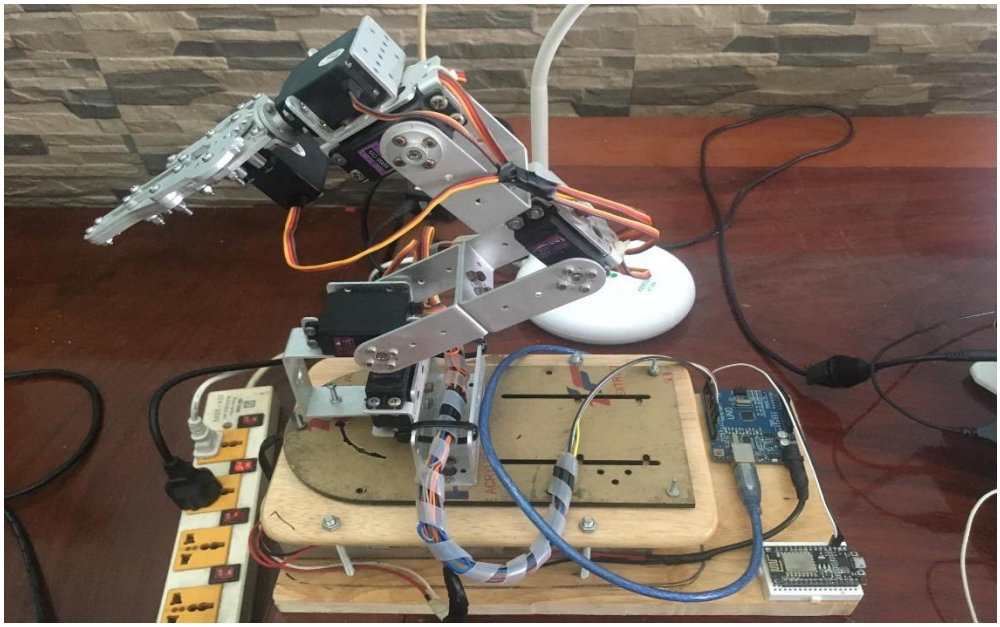


Figure 6: Fully Constructed Manipulator

3.3.1. Arduino Uno Board

The used microcontroller is open-source and is the third revised edition Arduino Uno board. In this version, the Arduino's main processing power and storage come from one ATmega328P microcontroller chip, of which the clock speed is 16 MHz. This chip has 32 KB of flash memory, 2 KB of static random-access memory, and 1 KB of electrically erasable programmable read-only memory. Power is provided via the built-in USB-B port or via a 6-12 V DC power source through the built-in female power pins or the power jack onboard. The USB-B port is also used to upload the compiled program into the Arduino through a computer. One 5 V voltage regulator is included in this model as well as one hardware reset button.

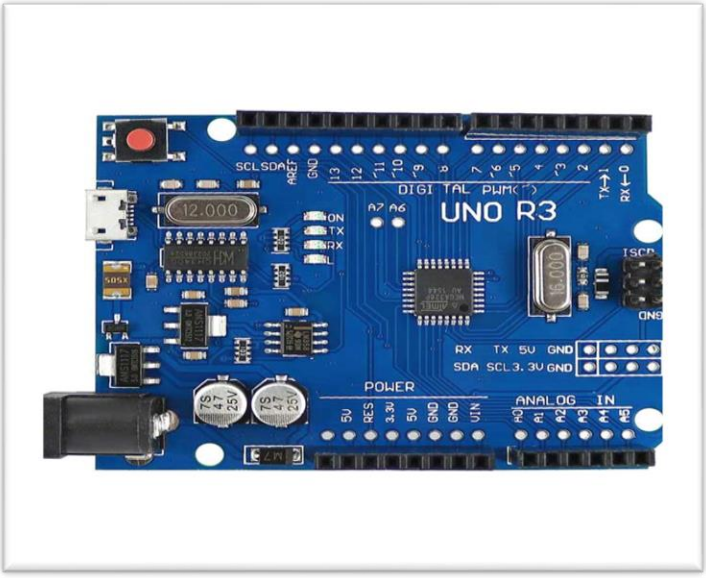


Figure 7: Arduino Uno R3 board

This version of the Arduino board has six female analog input pins, two of which are one I^2C data line (SDA) and one I^2C clock line (SCL) embedded on analog pin 4 and pin 5 respectively. Among the 14 digital I/O pins are 6 PWM-supported pins.[4]

3.3.2. Node MCU DEVKIT 1.0

With the size of 25x50 mm, using NodeMCU as an open-source Lua-based firmware, the board specially targets for IoT-based Applications. The firmware runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware that is based on the ESP-12 module.

This KIT is built based on Lua CP2102 and is designed to be compatible with Arduino IDE for programming and code uploading, making it a simple board to use. The board provides SPI, and I2C communication protocol, and especially uses the new and most stable UART communication and charging chip, CP2102, which is capable of self-recognizing Driver on all Windows and Linux operating systems since this is an enhanced version provided from versions using cheap refill IC CH340.

Furthermore, all GPIO pins are fully compatible with the NodeMCU firmware, and the power supply is 5VDC micro USB or Vin. However, GPIO communicates at 3.3VDC, so if any signal using more voltage than 3.3VDC can cause unexpected effect for the board.[5]

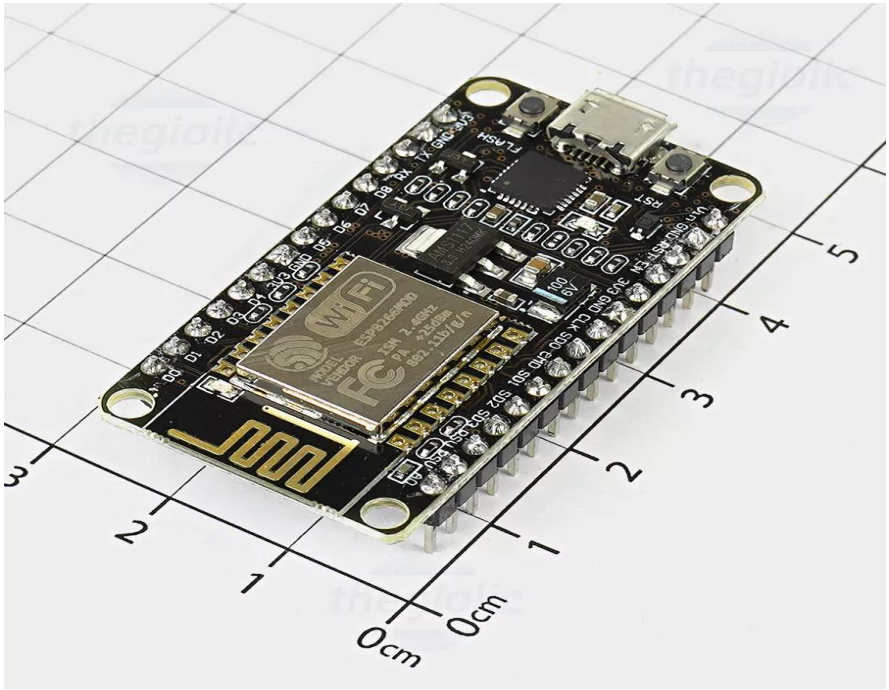


Figure 8: NodeMCU ESP8266 LUA CP2102

3.4. Software Design

This part includes the algorithm implemented inside the control sub-system. Furthermore, a model of HTML web server is mentioned, and how the web communicates with a lower layer using special methods.

3.4.1. Overall Design

The Arm moves when it receives signals from the Web Server, and the signals are transferred through a special transfer protocol combination including UART and LAN through a Wireless network:

- By clicking on any buttons on the Web Server, a user sends a signal to ESP through LAN using a wireless connection and TCP/IP (static).
- The signal then is transferred through the RX gate (UART) to Arduino R3.
- Then the signal is read by Arduino board as a char-type buffer.
- MCU then checks the condition of two cases: Uppercase letters and Lowercase letters, then increase or decrease the servo degree respectively.
- After that, MCU continues to check the Maximum and Minimum degree condition, then sets the servo degree to Max if the increased value is more than or equal to the maximum value, and Min if the decreased value is less than or equal to the minimum value.
- The pin using that buffer is identified.
- Corresponding Servos will then perform actions.[6]

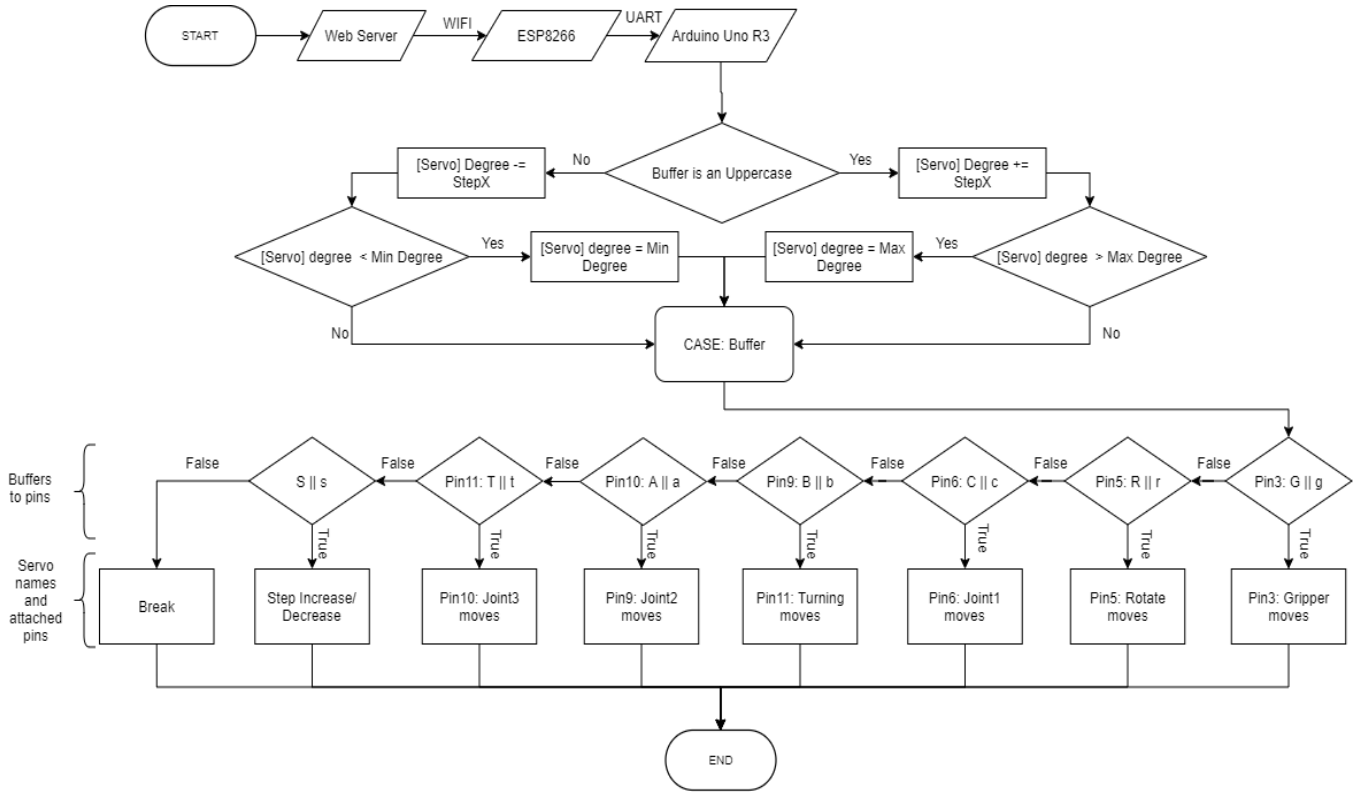


Figure 9: Flow chart

3.4.2. Communication Interface

The created web page is a friendly user interface. It allows users to control the Manipulator remotely via buttons on a virtual control panel. The web page was written in HTML, together with CSS and JavaScript, while the button symbols were chosen from font-awesome website. Bootstrap, one of the most famous frameworks (library) for designing websites and web applications, was also included at the beginning of the design process for better performance and easier coding of the user interface.[7]

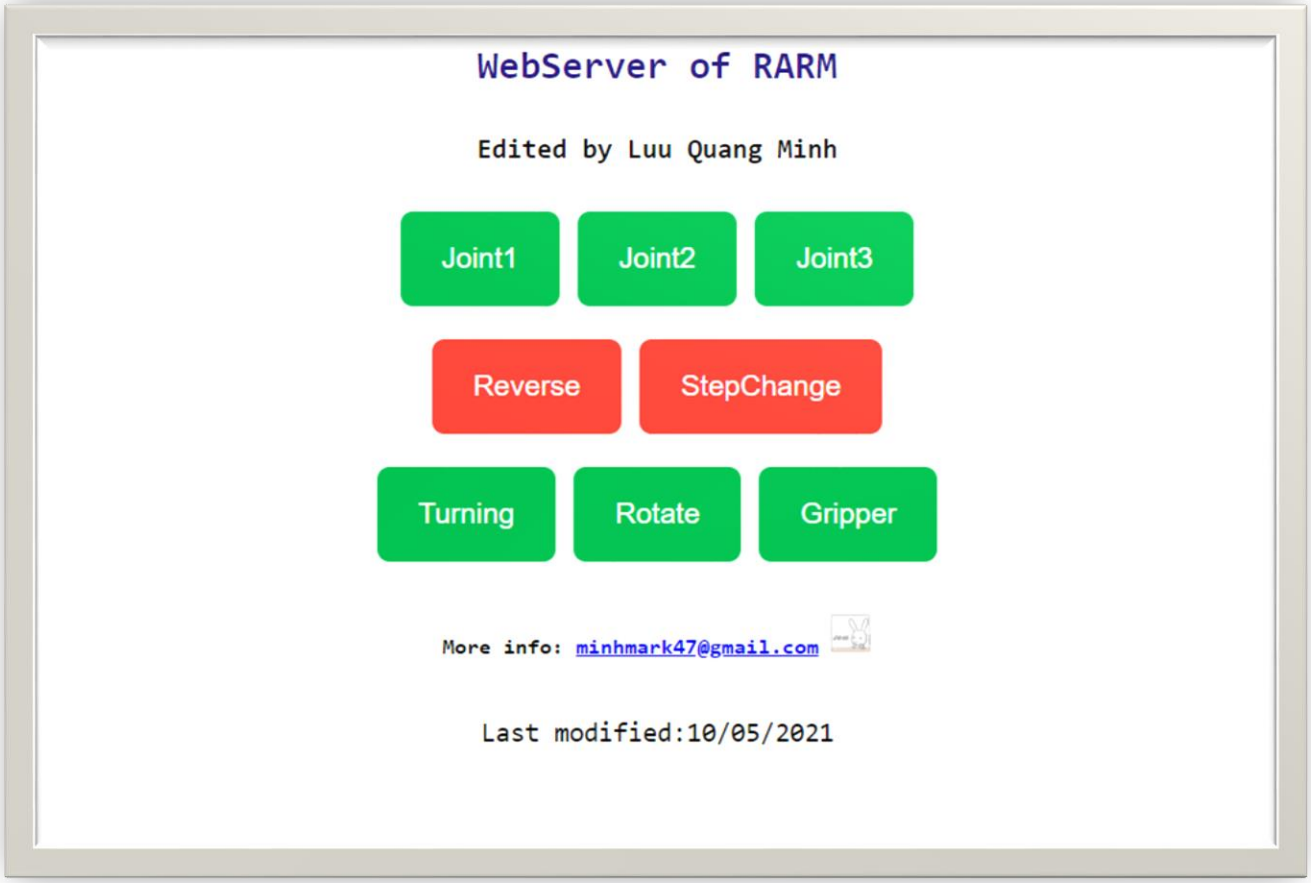


Figure 10: Web Serser at 192.168.1.5 static IP (UI)

3.4.3. Algorithm

The idea is to attach six servos to twelve char-type buffers (G||g, R||r, A||a, B||b, C||c, T||t), and the step (S||s) to increase/Decrease step from 1 to 10 degree of rotation. The uppercase letters are used for the increasing process, while the lowercase letters are used for decreasing process. Unsigned integer 8 bits (unint8_t) type is used for all variables to save the memory and is compatible with UART transmission. Therefore, packet scheme uses 8N1 (1 start bit, 8 data bits, 1 stop bit), and no parity bit check is included.

The algorithm is easy to implement, as it just increases or decreases the servo degree by the value of StepX (StepX={1,2,3,4,5,6,7,8,9,10}) whenever there is a buffer, after checking the Minimum and Maximum degrees of the servos through a special function type uint8_t (see Appendix C: Implemented code for Arduino).[8]

```
Algorithm: Servo control Algorithm
Input: x ϵ f, buffer that is sent
Output: servo
1       Initialize reverse_status = 0
2               f(i), set of buffers
3               h(i) = {T,A,B,C,R,G,S}
4               g(i) = {t,a,b,c,r,g,s}
5               binary servo[6] = {pin11,pin10,pin9,pin6,pin5,pin3} = 000000
6       if (reverse_status == 0) then
7               f(x) = h(x)
8       else
9               f(x) = g(x)
10      end
11      for i = 1 to 6 do
12              if (x = f(i)) then
13                      servo[i] = 1
14              else
15                      servo[i] = 0
16              end
17      end
18      return servo
```

Figure 11: Algorithm for controlling 6 DOF with char-type buffers

3.4.4. Web Server: GET and POST methods

A Web Server was created with HTML and posted based on some built-in ESP8266 libraries. This Web Server contains 8 buttons that have their function to control 6 DOF and step change. The Web Server uses basic HTTP request methods: 'GET and POST' to communicate through Wi-Fi in LAN, setting up a request-response protocol between client and server (client-server architecture). The ESP (client) submits an HTTP request to the Web page (server) and the server returns a response to the ESP, and the response containing status information about the request may also contain the requested content.

While HTTP GET is used to request data from a specified source, and often used to get value from APIs, HTTP POST is used to send data to a server to create/update a resource, and the data sent to the server with POST is stored in the request body of the HTTP request. Although it is more convenient to send a package in JSON format since the manipulator uses a char-type buffer as packets, each packet contains one buffer, the need for conversion between JSON and click-button javascript objects is unnecessary. Hence, all buffer can be transferred directly as plain text.[9]

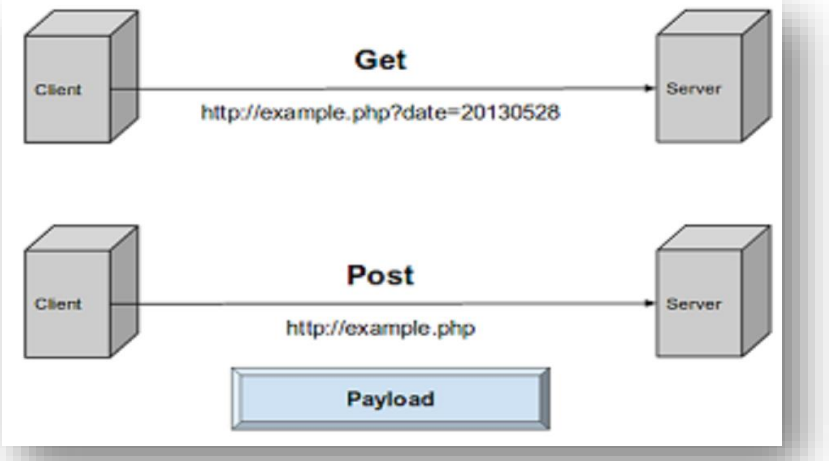


Figure 12: GET & POST methods

4.Conclusions

The reasons for using Arduino board for servo controlling instead of integrating all to ESP board are that the signal current at GPIO pins of ESP module is insufficient for servos to be activated, and receiving WIFI signal while processing and transferring data may lead ESP to overload and burn. Furthermore, ESP8266EX has 4 PWM output interfaces, and although they can be extended by users themselves, PWM interfaces are implemented via software programming, which means the realization by interruption of the timer and more space in ESP memory for some timer methods such as Overflow mode, CTC mode or Fast PWM mode.

Besides, only the GET method should be used in this project, and also only RX pin is used because the Arduino board is considered to be SLAVE which listens to ESP as MASTER all the time. The process does not use JSON format for wireless transferring objects since it sends a single packet that only contains a char-type buffer. In a nutshell, simplifying the wiring and process guarantees a faster response of the manipulator when controlled through a Web Server.

To summarize, this Robotic Arm can be operated by a Web Server and therefore can be controlled via the Internet. It can be used to lift an object as an experiment or for demonstration in some ECE modules. However, when the Robotic Arm moves, it shows no smooth movement and sometimes bounced, so further investigations and improvements should be carried out after this project.

5.References

[1] Towerpro MG996R Servo 25T. Retrieved from: https://www.towerpro.com.tw/product/mg996r/

[2] Frame description. Retrieved from: https://www.alibaba.com/product-detail/Aluminium-Robot-6-DOF-Arm-Clamp_60685999768.html

[3] Single Output Switchable Power Supply. Retrieved from: https://www.jameco.com/z/LRS-200-5-MEAN-WELL-200W-5V-40A-Single-Output-Switchable-Power-Supply_2219719.html

[4] Arduino UNO R3. Retrieved from: https://store.arduino.cc/usa/arduino-uno-rev3

[5] NodeMCU DEVKIT 1.0. Retrieved from: https://hshop.vn/products/kit-rf-thu-phat-wifi-esp8266

[6] Minh, L. Q. (2017). RARM report from phase 4 EEIT2017

[7] Wi-Fi controlled robot using NodeMCU. Retrieved from: https://www.iotdesignpro.com/projects/wifi-controlled-robot-using-esp12

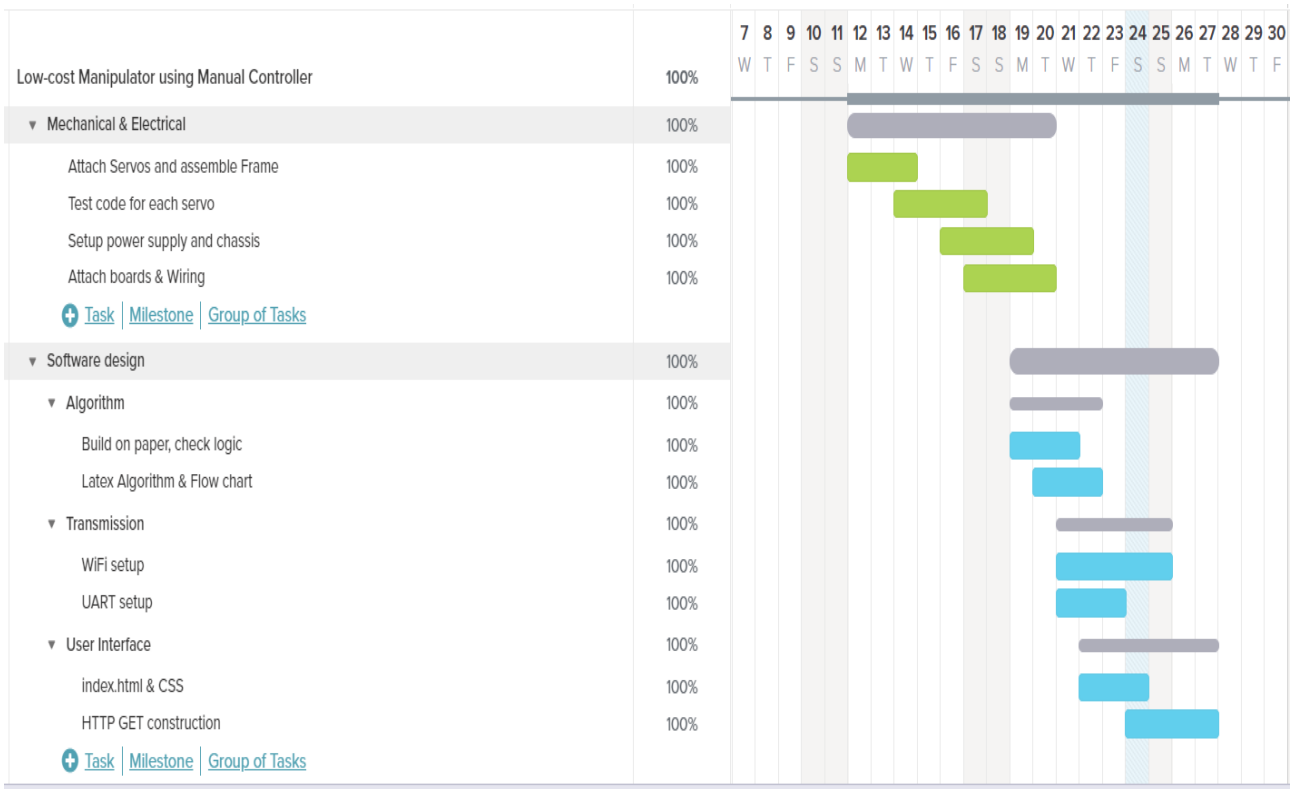
[8] PseudoCode Algorithm template. Retrieved from: https://www.overleaf.com/latex/examples/package-test-algorithm-slash-algorithmic/wxnhhlzmwfmq

[9] ESP8266 NodeMCU HTTP GET and HTTP POST with Arduino IDE (JSON, URL Encoded, Text). Retrieved from: https://randomnerdtutorials.com/esp8266-nodemcu-http-get-post-arduino

Appendix A: Glossary

Arduino	Opensource computer hardware and software project
Clock speed	Measured in Hertz, used to measure a processor's speed
DC	Direct Current
ECE	Electrical and Computer Engineering
ESP8266WifiSoC	Low-cost Wifi microchip with a full TCP/IP stack and microcontroller capability, produced by Espressif Systems in Shanghai, China[]
NodeMCU	Low-cost open source IoT platform
GND	Ground
UI	User Interface
FTP	File Transfer Protocol
RX	Transmit (RX mode)
HTTP	Hypertext Transfer Protocol
UART	Universal asynchronous receiver-transmitter
HTML	Hypertext Markup Language, used to create web pages
MCUs	Master Controlling Units
CSS	Cascading Style Sheets, used to modify the presentation of a document written in HTML
PWM	Pulse-width modulation
LAN	Local Area Network
HTML	Hypertext Markup Language
WIFI	A trademarked phrase that means IEEE 802.11x-popular wireless networking technology
Web Server	A program that uses HTML and some functional javascript code to serve the files that form Web pages to users, in response to their requests.
VCC	Power supply pin
VGU	Vietnamese – German University

Appendix B: Gantt Chart



Appendix C: Implemented code for Arduino

```
/*
    ____________________________________________________________________
    |                     Robotic Arm project (RARM)                    |
    |-------------------------------------------------------------------|
    |-----------------------Supervisor: Dr.Chan--------------------------|
    |---------------------Student: Lưu Quang Minh-----------------------|
    |-----------------Built for Senior project at VGU-------------------|
    |-------All Algorithms and Codes are created by Luu Quang Minh------|
    |-------------------------Updated on Github:-----------------------|
    |------https://github.com/minminlittleshrimp/RARM_senior_project-----|
    |--------------------Built for non-commercial use-------------------|
    |___________________________________________________________________|
*/
#include <stdint.h>
#include <Servo.h>

//Servo list
Servo Joint1,Joint2,Joint3;
Servo Gripper,Turning,Rotate;

char buff;

//Max & Min rotation
#define Joint1DH 140
#define Joint1DL 50

#define Joint2DH 160
#define Joint2DL 20

#define Joint3DH 160
#define Joint3DL 50

#define TurningDH 170
#define TurningDL 20

#define RotateDH 180
#define RotateDL 0

#define GripperDH 180
#define GripperDL 125

//-------------------------------------------------//

//Global Variable
uint8_t StepX=1;
uint8_t _DegJoint1=100,_DegJoint2=80,_DegJoint3=60;
uint8_t _DegGripper=30,_DegTurning=90,_DegRotate=90;

//Function for each Servo
//------------------Joint1-------------------------//
uint8_t __Joint1(uint8_t _TempDegJoint1)
{
  if(_TempDegJoint1>_DegJoint1)
  {
  _DegJoint1 += StepX;
```

```
//Max rotation
  if(_DegJoint1>Joint1DH)
  _DegJoint1=Joint1DH;
  Joint1.write(_DegJoint1);
  delay(10);
  return 2;
  }
  else if(_TempDegJoint1<_DegJoint1)
  {
  _DegJoint1 -= StepX;
//Min rotation
  if(_DegJoint1<Joint1DL)
  _DegJoint1=Joint1DL;
  Joint1.write(_DegJoint1);
  delay(10);
  return 1;
  }
  else {Joint1.write(_DegJoint1);return 0;}
}

//------------------Joint2-------------------------//
uint8_t __Joint2(uint8_t _TempDegJoint2)
{
  if(_TempDegJoint2>_DegJoint2)
  {
  _DegJoint2 += StepX;
//Max rotation
  if(_DegJoint2>Joint2DH)
  _DegJoint2=Joint2DH;
  Joint2.write(_DegJoint2);
  delay(20);
  return 2;
  }
  else if(_TempDegJoint2<_DegJoint2)
  {
  _DegJoint2 -= StepX;
//Min rotation
  if(_DegJoint2<Joint2DL)
  _DegJoint2=Joint2DL;
  Joint2.write(_DegJoint2);
  delay(20);
  return 1;
  }
  else {Joint2.write(_DegJoint2);return 0;}
}

//------------------Joint3-------------------------//
uint8_t __Joint3(uint8_t _TempDegJoint3)
{
  if(_TempDegJoint3>_DegJoint3)
  {
  _DegJoint3 += StepX;
//Max rotation
  if(_DegJoint3>Joint3DH)
  _DegJoint3=Joint3DH;
  Joint3.write(_DegJoint3);
  delay(10);
```

```
  return 2;
  }
  else if(_TempDegJoint3<_DegJoint3)
  {
  _DegJoint3 -= StepX;
//Min rotation
  if(_DegJoint3<Joint3DL)
  _DegJoint3=Joint3DL;
  Joint3.write(_DegJoint3);
  delay(10);
  return 1;
  }
  else {Joint3.write(_DegJoint3);return 0;}
}

//------------------Gripper-------------------------//
uint8_t __Gripper(uint8_t _TempDegGripper)
{
  if(_TempDegGripper>_DegGripper)
  {
  _DegGripper += StepX;
//Max rotation
  if(_DegGripper>GripperDH)
  _DegGripper=GripperDH;
  Gripper.write(_DegGripper);
  delay(10);
  return 2;
  }
  else if(_TempDegGripper<_DegGripper)
  {
  _DegGripper -= StepX;
//Min rotation
  if(_DegGripper<GripperDL)
  _DegGripper=GripperDL;
  Gripper.write(_DegGripper);
  delay(10);
  return 1;
  }
  else {Gripper.write(_DegGripper); return 0;}
}

//------------------Turning-------------------------//
uint8_t __Turning(uint8_t _TempDegTurning)
{
  if(_TempDegTurning>_DegTurning)
  {
  _DegTurning += StepX;
//Max rotation
  if(_DegTurning>TurningDH)
  _DegTurning=TurningDH;
  Turning.write(_DegTurning);
  delay(10);
  return 2;
  }
  else if(_TempDegTurning<_DegTurning)
  {
  _DegTurning -= StepX;
```

```
//Min rotation
  if(_DegTurning<TurningDL)
  _DegTurning=TurningDL;
  Turning.write(_DegTurning);
  delay(10);
  return 1;
  }
  else {Turning.write(_DegTurning); return 0;}
}

//------------------Rotate-------------------------//
uint8_t __Rotate(uint8_t _TempDegRotate)
{
  if(_TempDegRotate>_DegRotate)
  {
  _DegRotate += StepX;
//Max rotation
  if(_DegRotate>RotateDH)
  _DegRotate=RotateDH;
  Rotate.write(_DegRotate);
  delay(10);
  return 2;
  }
  else if(_TempDegRotate<_DegRotate)
  {
  _DegRotate -= StepX;
//Min rotation
  if(_DegRotate<RotateDL)
  _DegRotate=RotateDL;
  Rotate.write(_DegRotate);
  delay(10);
  return 1;
  }
  else {Rotate.write(_DegRotate);return 0;}
}

//----------------Main code for control 6 servos--------------//
void setup()
{
  Serial.begin(9600);
  Serial.println("D6ARMv01");

//Set servos to pins

  Turning.attach(11);
  Joint3.attach(10);
  Joint2.attach(9);
  Joint1.attach(6);
  Rotate.attach(5);
  Gripper.attach(3);

//Set all check-variable-->.write(); return 0;

  __Joint1(70);__Joint2(70);__Joint3(70);
  __Gripper(130);__Turning(90); __Rotate(90);
}
```

```
void loop()
{

//Variable to increase by StepX in rotation
  uint8_t TempJoint1,TempJoint2, TempJoint3;
  uint8_t TempGripper, TempTurning, TempRotate;
  uint8_t StepJoint1, StepJoint2, StepJoint3;
  uint8_t StepGripper, StepTurning, StepRotate;

  if(Serial.available()>0)
  {
    buff=Serial.read();
    Serial.print(buff);
    Serial.print(">>");
    switch(buff)
    {
    //A and a: control Joint1

    case'A':
    StepJoint1=_DegJoint1+StepX;
    TempJoint1=__Joint1(StepJoint1);
    Serial.print("+Deg:");Serial.print(_DegJoint1);
    Serial.print(" S:");Serial.println(TempJoint1); break;

    case'a':
    StepJoint1=_DegJoint1-StepX;
    TempJoint1=__Joint1(StepJoint1);
    Serial.print("-Deg:");Serial.print(_DegJoint1);
    Serial.print(" S:");Serial.println(TempJoint1); break;

    //B and b: control Joint2

    case'B':
    StepJoint2=_DegJoint2+StepX;
    TempJoint2=__Joint2(StepJoint2);
    Serial.print("+Deg:");Serial.print(_DegJoint2);
    Serial.print(" S:");Serial.println(TempJoint2); break;

    case'b':
    StepJoint2=_DegJoint2-StepX;
    TempJoint2=__Joint2(StepJoint2);
    Serial.print("-Deg:");Serial.print(_DegJoint2);
    Serial.print(" S:");Serial.println(TempJoint2); break;

    //C and c: control Joint3

    case'C':
    StepJoint3=_DegJoint3+StepX;
    TempJoint3=__Joint3(StepJoint3);
    Serial.print("+Deg:");Serial.print(_DegJoint3);
    Serial.print(" S:");Serial.println(TempJoint3); break;

    case'c':
    StepJoint3=_DegJoint3-StepX;
    TempJoint3=__Joint3(StepJoint3);
    Serial.print("-Deg:");Serial.print(_DegJoint3);
    Serial.print(" S:");Serial.println(TempJoint3); break;
```

```
    //G and g: control Gripper

    case'G':
    StepGripper=_DegGripper+StepX;
    TempGripper=__Gripper(StepGripper);
    Serial.print("+Deg:");Serial.print(_DegGripper);
    Serial.print(" S:");Serial.println(TempGripper); break;

    case'g':
    StepGripper=_DegGripper-StepX;
    TempGripper=__Gripper(StepGripper);
    Serial.print("-Deg:");Serial.print(_DegGripper);
    Serial.print(" S:");Serial.println(TempGripper); break;

    //T and t: control Turning

    case'T':
    StepTurning=_DegTurning+StepX;
    TempTurning=__Turning(StepTurning);
    Serial.print("+Deg:");Serial.print(_DegTurning);
    Serial.print(" S:");Serial.println(TempTurning); break;

    case't':
    StepTurning=_DegTurning-StepX;
    TempTurning=__Turning(StepTurning);
    Serial.print("-Deg:");Serial.print(_DegTurning);
    Serial.print(" S:");Serial.println(TempTurning); break;

    //R and r: control Rotate

    case'R':
    StepRotate=_DegRotate+StepX;
    TempRotate=__Rotate(StepRotate);
    Serial.print("+Deg:");Serial.print(_DegRotate);
    Serial.print(" S:");Serial.println(TempRotate); break;

    case'r':
    StepRotate=_DegRotate-StepX;
    TempRotate=__Rotate(StepRotate);
    Serial.print("-Deg:");Serial.print(_DegRotate);
    Serial.print(" S:");Serial.println(TempRotate); break;

    case'S':
    StepX++;
    if(StepX>10) StepX=10;
    break;

    case's':
    StepX--;
    if(StepX<1) StepX=1;
    break;

    default: break;
    }
  }
}
```

Appendix D: HTML.index & Wifi tracking code for ESP8266

```
#include <ESP8266WiFi.h>
#include <ESPAsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <SoftwareSerial.h>

char buff;
bool reverse_status = 0;
SoftwareSerial uart_gate(3,1);

const char* ssid = "A,Son";
const char* password = "minhhieu1130";


// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>

<head>
  <title>RARM control WebServer</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <style>
  html {
    font-family: Consolas;
    display: inline-block;
    margin: 0px auto;
    text-align: center;
  }
  h3 {
    color: #1e0f76;
    padding: 0vh;
  }
  p {
    font-size: 1rem;
  }
  .button {
    display: inline-block;
    background-color: #00ba4d;
    border: none;
    border-radius: 6px;
    color: white;
    padding: 15px 20px;
    text-decoration: none;
    font-size: 14px;
    margin: 0px auto;
    cursor: pointer;
  }
  .button_stop {
    background-color: #f44336;
  }

   </style>
</head>
```

```
<body>
  <h3>WebServer of RARM</h3>
  <h5>Edited by Luu Quang Minh</h5>
  <p><a href="/Joint1"><button class="button button1">Joint1</button></a>
    <a href="/Joint2"><button class="button button2">Joint2</button></a>
    <a href="/Joint3"><button class="button button3">Joint3</button></a></p>
  <p><a href="/Reverse"><button class="button button_stop">Reverse</button></a>
    <a href="/StepChange"><button class="button
button_stop">StepChange</button></a></p>
  <p><a href="/Turning"><button class="button button4">Turning</button></a>
    <a href="/Rotate"><button class="button button5">Rotate</button></a>
    <a href="/Gripper"><button class="button button6">Gripper</button></a></p>

<div class="footer">
    <div class="container">
        <div class="row">
            <div class="col-md-12">
                <h6> More info: <a
href="mailto:minhmark47@gmail.com">minhmark47@gmail.com</a>
                    <a
href="https://www.facebook.com/profile.php?id=100004186217509">
                        <img src="img/avatar.jpg" alt="Facebook account"
width="20" height="20"></a></h6>
                <tt>Last modified:10/05/2021</tt>
            </div>
        </div>
    </div>
</div>
</body>

</html>)rawliteral";

void setup(){
  Serial.begin(115200);
  uart_gate.begin(9600);
  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("..");
  }
  Serial.println(WiFi.localIP());

  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html);
  });

  server.on("/Joint1", HTTP_GET, [](AsyncWebServerRequest *request){
    if(reverse_status == 0) uart_gate.write('a');
    else uart_gate.write('A');
    request->send(200, "text/html", index_html);
  });

  server.on("/Joint2", HTTP_GET, [](AsyncWebServerRequest *request){
    if(reverse_status == 0) uart_gate.write('b');
```

```
    else uart_gate.write('B');
    request->send(200, "text/html", index_html);
  });

  server.on("/Joint3", HTTP_GET, [](AsyncWebServerRequest *request){
    if(reverse_status == 0) uart_gate.write('c');
    else uart_gate.write('C');
    request->send(200, "text/html", index_html);
  });

  server.on("/Reverse", HTTP_GET, [](AsyncWebServerRequest *request){
    if(reverse_status == 0) reverse_status = 1;
    else reverse_status = 0;
    request->send(200, "text/html", index_html);
  });

  server.on("/StepChange", HTTP_GET, [](AsyncWebServerRequest *request){
    if(reverse_status == 0) uart_gate.write('s');;
    else uart_gate.write('S');
    request->send(200, "text/html", index_html);
  });

  server.on("/Turning", HTTP_GET, [](AsyncWebServerRequest *request){
    if(reverse_status == 0) uart_gate.write('t');
    else uart_gate.write('T');
    request->send(200, "text/html", index_html);
  });

  server.on("/Rotate", HTTP_GET, [](AsyncWebServerRequest *request){
    if(reverse_status == 0) uart_gate.write('r');
    else uart_gate.write('R');
    request->send(200, "text/html", index_html);
  });

  server.on("/Gripper", HTTP_GET, [](AsyncWebServerRequest *request){
    if(reverse_status == 0) uart_gate.write('g');
    else uart_gate.write('G');
    request->send(200, "text/html", index_html);
  });
  // Start server
  server.begin();
}
void loop(){}
```