Remote Control of a Humanoid Robot Arm

H. Pérez Pérez

University CEU San Pablo, Madrid, Spain, h.perez10@usp.ceu.es.

Abstract

This project presents the implementation of a remote-control interface for a 3D-printed humanoid arm designed by Youbonic. The focus of this project was setting up a real-time remote-control interface using Blynk, an IoT platform that provides a web interface that allows for remote control of IoT devices, for the robotic arm and allows the microcontroller to connect through the internet to it. Blynk's interface can easily be used to execute a series of predefined gestures such as the one used on this project demo, pointing with the index finger.

1. Introduction

The emergence of remote operation within the Internet of Things (IoT) showcases a substantial potential for enhancing a diverse array of fields, spanning from intricate surgical procedures and hazardous industrial settings to extraterrestrial exploration and disaster response initiatives. Great examples of such use cases in the medical field would be the remote operation of the da Vinci® Surgical System [1], allowing a specialist to perform operations without having to be physically in the operating room; or providing more advanced medical support in rural areas, such as performing a remote echography [2]. Another great example within the medical field is the CyberKnife[3], a device capable of performing stereotactic surgery (Radiation therapy mainly used to slow the growth or destroy tumors that are unreachable through conventional surgery).

Remote-controlled arms enable a multitude of practical applications across other fields, such as the scientific and emergency response fields. One notable example in the scientific field is the development of telelaboratories [4], where these remotely operated labs offer their services through web interfaces, enabling researchers to conduct experiments and manipulate the remote environment from a distance, using multiple ways of interaction. In the mentioned example, the system architecture permits any external program to have access to almost every feature of the telelaboratory, allowing for the design of experiments that can be carried out either remotely or in person. In the context of emergency response, these arms play a crucial role, especially in scenarios like bomb disposal [5], where their remote operation proves essential in averting potential harm to human lives by providing a safe and controlled means of handling dangerous materials and situations.

Another field in which robotic arms can be used is the education field, serving as a platform in which to develop and test out new concepts. In this context, the main goal is to implement a web interface facilitating remote control of a humanoid robotic arm, a 3-D printed arm designed by Youbionics [6], configured in a previous project [7]. The remote operation of this arm is achieved through the integration of IoT technologies [8]: the robotic arm is controlled through Blynk [9], an IoT platform explicitly prepared to provide web interfaces and cloud infrastructure for IoT devices.

2. Hardware

The arm design employed in this project is the same as the one used in the previous iteration of this project. It consists of a complete 3D-printed humanoid arm, whose schematics were obtained from Youbionic. This humanoid arm (as shown in Figure 1) is formed by shoulder, elbow, forearm, wrist, and hand.

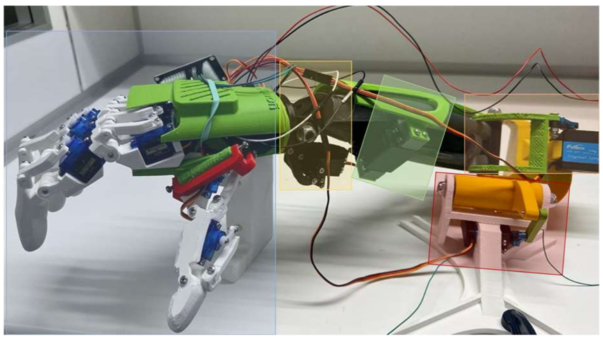


Figure 1. *Humanoid Arm used on the project. Blue: hand and fingers; Yellow: wrist articulation; Green: Forearm; Orange: elbow articulation; Red: shoulder articulation.*

The whole arm is operated by a total of 15 servos: One for each of the main arm articulations (wrist, forearm, biceps, and shoulder), providing the arm with four degrees of freedom, as each servo provides motion in a single plane; and another 11 servos for hand movement. In the instance of the thumb, it operates via three servos, with two facilitating the flexing motion of the finger, and the third serving as the palm servo (abduction and adduction movements). Conversely, each remaining finger achieves flexion through two servos, mirroring the thumb's arrangement. This implies that, akin to the arm servos, all finger servos permit motion within a single plane, resulting in two degrees of freedom for each finger. Notably, the thumb stands out with three degrees of freedom because it is actuated with an additional servo.

2.1. Background

This project is the continuation of a previous project where the arm was calibrated and configured for in-person use. The prior project placed a central emphasis on realizing targeted arm movements, specifically, the action of pressing down a button (illustrated in Figure 2) and refining the imperfections found in the 3D-printed components.

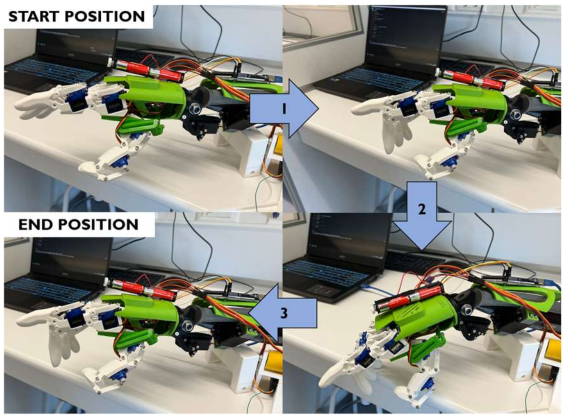


Figure 2. Depiction of the three movements (1→2 →3) that add up to form the desired gesture.

2.2. Proposal

As mentioned earlier, the preceding project did not incorporate remote control capabilities, as it did not consider the remote operation of the arm within its scope. This limitation stemmed from the utilization of a main microcontroller, an Arduino board (Arduino Mega 2560 R3), lacking a Wi-Fi module. Consequently, once the code was uploaded onto the board, it became fixed, with no option for alteration without access to a computer capable of loading code into an Arduino board.

To solve this limitation, the previous controller had to be replaced with a new microcontroller with support for Wi-Fi connection. A NodeMCU Lua Lolin V3 (Figure 3) was selected to replace the previous Arduino Mega 2560.

Figure 3. NodeMCU board with integrated ESP8266 Wi-Fi module.

The use of the "ESP8266 Core for the Arduino IDE" [10] allowed for the seamless reuse and integration of the previous code [11] into the NodeMCU board utilized in this project, significantly streamlining the software setup process for controlling the arm in the current project.

The hardware architecture remains the same as with the previous project: The main microcontroller connects to a PWM servo driver board, which generates the digital PWM signals, according to the microcontroller directives, used to control the servo position (Figure 4).

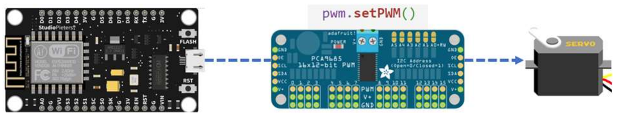


Figure 4. Through the PWM board, the servos can be controlled using the function setPWM(); included in Adafruit's library.

The arm is actuated using the following components:

- NodeMCU Lua Lolin V3 (ESP8266 MOD 12-F)
- Adafruit PWM/Servo driver board (PCA9685)
- 11 Micro Servos SG90 (finger and palm servos)
- 2 FeeTech Digital Servos FS5323M (wrist and forearm joint)
- 2 FeeTech 15kg.cm Servos (elbow and shoulder joint)

In order to control the arm through Blynk's web interface, an internet connection to Blynk servers is required from the machine used to access the web interface. Then, the arm servos are controlled through a combination of a NodeMCU board (Lolin Lua V3) and an Adafruit 12-bit PWM/Servo driver board (refer to Figure 5).

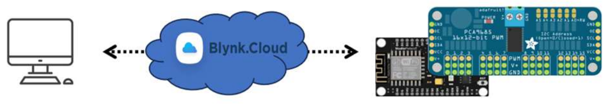


Figure 5. Remote control diagram of the arm.

3. Integration

The primary objective of this project is to enable remote control of a robotic arm (specifically, Youbionic's humanoid arm) through the Internet, allowing for remote operation from anywhere as long as an internet connection is available. To achieve this, the initial step involved integrating Blynk connectivity into the code, facilitating the connection of our controller to Blynk servers, and enabling remote function execution. Subsequently, the code underwent modifications to provide simple and extensive functionality through the Blynk web interface. Finally, the code recycled from the previous project was adapted to incorporate positional control for each servo, enabling the arm to store and recall the current position of each servo.

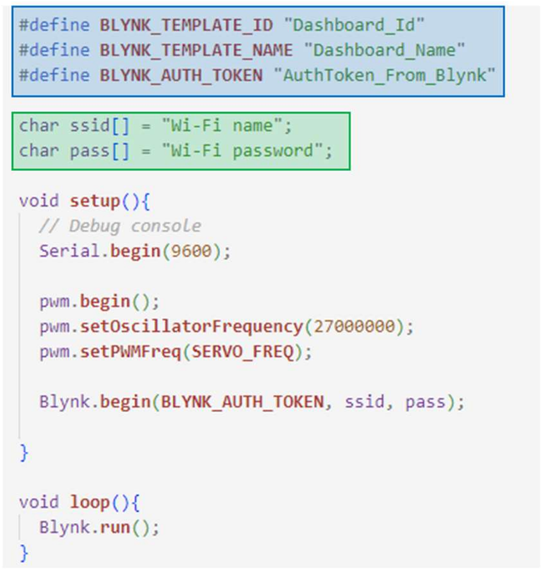
3.1. Blynk Connection

In the previous project, the Arduino Mega 2560 R3 served as the controller. However, due to the internet connection requirement of this project, it was necessary to find a solution to connect the Arduino, a microcontroller lacking a Wi-Fi module, to the Internet.

Two options were considered: the first involved attempting a remote connection through a USB serial connection. This would necessitate linking the Arduino board to a computer running a now deprecated Blynk app, designed for boards lacking Wi-Fi modules. While this option would have provided a straightforward solution, the necessity for a USB connection to a computer rendered it impractical for biomedical applications of the arm. It also proved unfeasible due to the lack of support for this connection method in the current version of Blynk.

The second option, chosen for this project, involved replacing the controller with one capable of internet connectivity. The selected replacement was a NodeMCU Lolin Lua v3 (ESP8266-12F). This board could connect to the internet through the ESP8266 module, and with support from the ESP8266 Core for the Arduino IDE, it is configurable like any other Arduino board using the Arduino IDE.

To use Blynk servers and interface, Blynk provides libraries and code templates that substitute the normal code in our MCU (Refer to Figure 6). These code templates require the modification of specific parameters (Wi-Fi credentials, Blynk dashboard info) according to the ones in use for the project: The Wi-Fi credentials (ssid, and pass) are those of the Wi-Fi the board will connect to; while the template_id, template_name and auth_token refer to the Blynk dashboard our MCU will be controlled from.



```
#define BLYNK_TEMPLATE_ID "Dashboard_Id"
#define BLYNK_TEMPLATE_NAME "Dashboard_Name"
#define BLYNK_AUTH_TOKEN "AuthToken_From_Blynk"

char ssid[] = "Wi-Fi name";
char pass[] = "Wi-Fi password";

void setup(){
  // Debug console
  Serial.begin(9600);

  pwm.begin();
  pwm.setOscillatorFrequency(27000000);
  pwm.setPWMFreq(SERVO_FREQ);

  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);

}

void loop(){
  Blynk.run();
}
```

Figure 6. *Code required for the NodeMCU board connection to Blynk and servo control. Blue highlight: Dashboard data provided by Blynk; Green highlight: Wi-Fi network identification.*

The connection between the NodeMCU board and the Servo driver board, an Adafruit PCA9685 board (PWM Servo driver), is performed through the I2C interface. According to the documentation provided by the seller [12] from which the NodeMCU was acquired, the pins that should correspond to the I2C channels are pin D5 - GPIO14(SCL) and pin D4 - GPIO2(SDA). However, this pin setup doesn't work with the NodeMCU board used (even though it is the documentation of said NodeMCU board), and after investigating, the correct pins were found: pin D1 - GPIO5 (SCL) and pin D2 - GPIO4 (SDA). The final wiring connection between both boards is depicted in Figure 7.

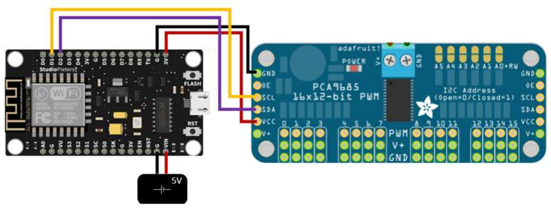
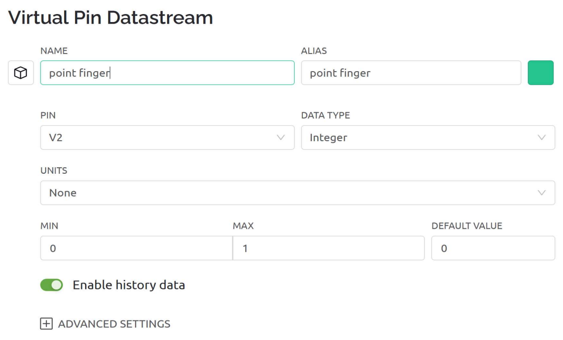


Figure 7. *Diagram of the I2C connection between the NodeMCU board and the Adafruit PCA9685.*

3.2. Blynk Dashboard setup

Blynk's interface facilitates the creation of virtual pins (Figure 8), enabling the board to read from or write to them for data exchange over an internet connection.

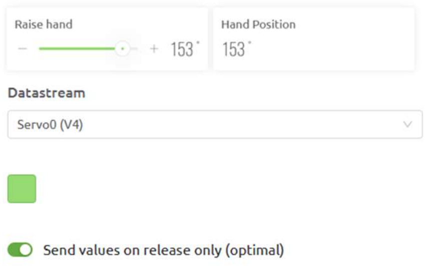


Virtual Pin Datastream

NAME: point finger
ALIAS: point finger
PIN: V2
DATA TYPE: Integer
UNITS: None
MIN: 0
MAX: 1
DEFAULT VALUE: 0
Enable history data
ADVANCED SETTINGS

Figure 8. *Configuration panel of each Virtual Pin.*

Accessible from the web interface, these virtual pins permit remote monitoring and management of values from any device with an internet connection. The Blynk dashboard provides configuration options to determine the frequency of information transmission to the board, allowing for choices between sending data on release or continuously. (Figure 9).



Raise hand 153°
Hand Position 153°

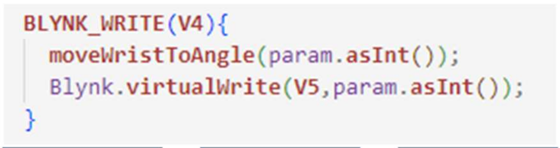
Datastream
Servo0 (V4)

Send values on release only (optimal)

Figure 9. *Top image: Blynk dashboard; Bottom image: Configuration of the "Raise hand" widget of the Dashboard.*

For the board to interact with Blynk servers and exchange information, it's necessary to utilize the provided functions "BLYNK_WRITE(**VirtualPin**)" and "Blynk.virtualWrite(**VirtualPin**, value)". The former is invoked each time a value is transmitted from the Blynk dashboard, with the caveat that only the function whose **VirtualPin** aligns with the configured virtual pin of the modified widget will be called. The value can be retrieved from the **param** object using **param**.asInt() (assuming the expected value is an integer). In order to send information, the second function is used, requiring the specification of the virtual pin to be modified, followed by the desired value to be set.

In the code snippet provided (see Figure 10), functionality is integrated into both widgets, namely Raise Hand and Hand Position, within the pre-existing dashboard (as depicted in Figure 9). The virtual pin associated with the Raise Hand widget (V4) is read, and the obtained value is used to adjust the hand to the specified angle. Subsequently, this value is written to virtual pin V5, linked to the Hand Position widget, thereby updating the hand's position value on the dashboard.



```
BLYNK_WRITE(V4){
  moveWristToAngle(param.asInt());
  Blynk.virtualWrite(V5,param.asInt());
}
```

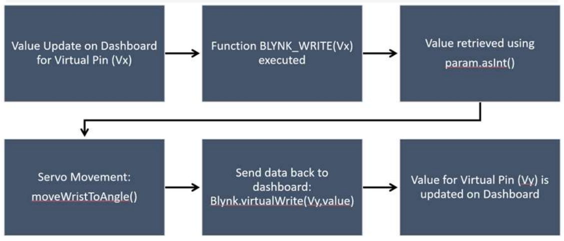
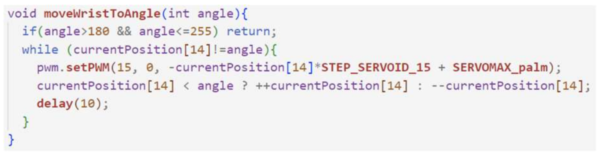

Value Update on Dashboard for Virtual Pin (Vx)
Function BLYNK_WRITE(Vx) executed
Value retrieved using param.asInt()
Servo Movement: moveWristToAngle()
Send data back to dashboard: Blynk.virtualWrite(Vy,value)
Value for Virtual Pin (Vy) is updated on Dashboard

Figure 10. *Top: Function receiving a value from Blynk's dashboard and sending data back. Bottom: Function diagram.*

3.3. Software Improvements

The code from the earlier project was designed for basic gestures, lacking positional control for the servos. This limitation was addressed by incorporating an array of 15 bytes, storing the current angle of all 15 servos after each gesture (if the angle was modified). With this revised configuration [13], during a movement, the arm initiates from its current position instead of resetting to a starting position before executing the desired movement (Figure 11).



```
void moveWristToAngle(int angle){
  if(angle>180 && angle<=255) return;
  while (currentPosition[14]!=angle){
    pwm.setPWM(15, 0, -currentPosition[14]*STEP_SERVOID_15 + SERVOMAX_palm);
    currentPosition[14] < angle ? ++currentPosition[14] : --currentPosition[14];
    delay(10);
  }
}
```

Figure 11. *Function that moves the wrist to a desired angle.*

4. Conclusions

This project successfully achieved its goal by utilizing a NodeMCU board in conjunction with Blynk, enabling remote arm control through Blynk's user-friendly and fast-to-set-up web interface, as demonstrated in the project showcase [14].

However, the Blynk interface poses a restriction when controlling the arm, making individual control of the arm servos a laborious task. This limitation can be circumvented by using predefined gestures, removing the individual servo control from Blynk's interface, and leaving it as a gesture selector. Predefined gestures avoid Blynk's interface problem by having the parameters required for each gesture locally stored, meaning that the end user doesn't have to adjust each servo individually, simplifying the experience.

A significant enhancement for this project could involve the development of a standalone application dedicated to remote arm control. While Blynk provides a viable solution, it has its limitations. Achieving precise control through Blynk's interface, as previously mentioned, would be a tedious process. It involves implementing flow controls to enable/disable immediate code execution. These controls, combined with sliders for each servo, would allow for maneuvering the arm into any conceivable position via Blynk's web interface without requiring code alterations.

The primary drawback with Blynk lies in both compatibility and reliability. As a private company, Blynk has the authority to discontinue support for a specific controller type at its discretion. An illustrative example is the cessation of support for devices utilizing the USB serial platform to connect to Blynk, a solution that is no longer viable for using Blynk's interface. Moreover, Blynk is designed as a web interface, which, while convenient for human use through its webpage, makes the solution nearly impossible to integrate with any other independently developed program. This characteristic makes Blynk incompatible with any alternative software that could potentially be employed to control the device.

References

[1] F. Richter, E. K. Funk, W. Seo Park, R. K. Orosco and M. C. Yip, "From Bench to Bedside: The First Live Robotic Surgery on the dVRK to Enable Remote Telesurgery with Motion Scaling," *2021 International Symposium on Medical Robotics (ISMR)*, Atlanta, GA, USA, 2021, pp. 1-7, doi: 10.1109/ISMR48346.2021.9661536.

[2] Kurt Boman, Mona Olofsson, Johan Forsberg, and Sven-Åke Boström.Remote-Controlled Robotic Arm for Real-Time Echocardiography: The Diagnostic Future for Patients in Rural Areas?. Telemedicine and e-Health. Mar 2009.

[3] Kilby, W., Naylor, M., Dooley, J.R., Maurer Jr, C.R. and Sayeh, S., 2020. A technical overview of the CyberKnife system. Handbook of robotic and image-guided surgery.

[4] R. Marin, P. J. Sanz, P. Nebot and R. Wirz, "A multimodal interface to control a robot arm via the web: a case study on remote programming," in *IEEE Transactions on Industrial Electronics*, vol. 52, no. 6, pp. 1506-1520, Dec. 2005, doi: 10.1109/TIE.2005.858733.

[5] Narayanan, S. and Reddy, C.R., 2015. Bomb defusing robotic arm using gesture control. *International Journal of Engineering Research and Technology*, *4*(02).

[6] "https://www.youbionic.com/". Youbionic. (Consulted: January 2024)

[7] H. Pérez Pérez. 2023. Control of a Humanoid Robot Arm.

[8] S. Fu and P. C. Bhavsar, "Robotic Arm Control Based on Internet of Things," 2019 IEEE Long Island Systems, Applications and Technology Conference (LISAT), Farmingdale, NY, USA, 2019, pp. 1-6, doi: 10.1109/LISAT.2019.8817333.

[9] Seneviratne, P., 2018. Hands-On Internet of Things with Blynk: Build on the power of Blynk to configure smart devices and build exciting IOT projects. Packt Publishing.

[10] "ESP8266 core for Arduino". GitHub. (Consulted: December 2023)

[11] "proyectos2_HumanoidArm". GitHub. (Consulted: December 2023)

[12] "NodeMCU Lua Lolin V3 Module ESP8266 ESP-12F". Az-delivery. (Consulted: December 2023)

[13] "proyectos3_RemoteArmControl". GitHub. (Consulted: December 2023)

[14] "Remote Operation of a Humanoid Robot Arm". Github Pages (Consulted: December 2023)