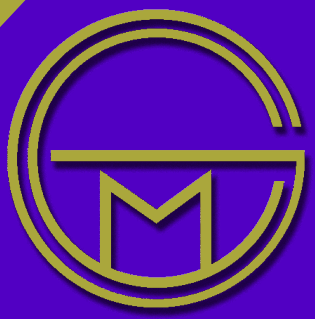
Team Project Report



Tomislav Novosel

r1036223

Skills Integration Lab 1

24.05.2025.

Contents

CV & Cloud Controlled Robotic Hands

YouTube DEMO: https://www.youtube.com/watch?v=kYk10_oYDqU

Team composition

As I have an interesting schedule due to me combining my career with the studies, I have asked and been allowed to be alone in the team. Main reason for that is the fact that my work is during the weekends in another country, and weekends are usually the main times while other students work on projects together, so it would be very problematic to align as I would mostly be unavailable when other are available.

Idea

The primary idea was to make a complete assembly kit with all the parts needed to build it for beginner IoT or AI pupils to make learning fun and engaging. They would have a chance to try Python coding, SBCs, PWM control, servo motors, robotics basics, computer vision and networking.

Another value is a proof of concept for a remote controlled robotic hand that can be used for example in an unreachable or hazardous environment from a remote and safe location.

I made two different hands to show different possibilities and possible interfaces. One involves simply connecting the hand to a laptop via USB while the other is connected to the Internet via WiFi and takes values from a MQTT broker.

Each finger is individually controlled by a small servo motor. One version opens and closes fingers, while the USB one has a more layered level of control, gradual, sensitive and with very low latency.

I used Python, MediaPipe libraries, servo motors, self-made 3d models, Arduino, Pico, OrangePi, a touchscreen and the Ubidots dashboard.

Both hands are iterations of a personal hobby project that gets more advanced with each iteration, and the plan is to develop as far as possible with regards to movement axis, range of motion, strength, control and ease of use.

Information-Value loop

COMPUTER VISION-BASED ROBOTIC HAND (VIA MEDIAPIPE + USB SERIAL)

A. DATA ACQUISITION (SENSE):

- Webcam captures real-time hand image
- MediaPipe Hands processes frame, detects 21 landmarks

B. DATA PROCESSING (THINK):

- PC script interprets landmark positions, calculates positions of fingers regarding to difference between pixels on the screen between landmarks
- Converts this to servo angles (0–180), using normalizing math functions

C. DATA COMMUNICATION (TRANSFER):

- Angles sent over USB serial to the Arduino
- Each servo is connected to indvidual pin and controled directly and live via PyFirmata

D. ACTUATION (ACT):

- Arduino recieves signals from PyFirmata
- Sends PWM to servos → robotic hand mimics real hand

E. FEEDBACK / VALUE:

- Real-time mirroring of user's hand
- High responsiveness with direct visual feedback
- Fun, POC

UBIDOTS IOT-CONTROLLED ROBOTIC HAND (VIA ORANGE PI + MQTT)

A. DATA ACQUISITION (SENSE):

- User taps a gesture or toggles fingers via Orange Pi touchscreen (Tkinter UI)

B. DATA PROCESSING (THINK):

- Orange Pi:
  - Maps gestures to per-finger servo angles
  - Prepares JSON payloads

C. DATA COMMUNICATION (TRANSFER):

- Orange Pi publishes servo angles (0–180) for:
  - /thumb, /index, /middle, /ring, /pinkie
- MQTT broker = Ubidots
- Pico W subscribes to these topics

D. ACTUATION (ACT):

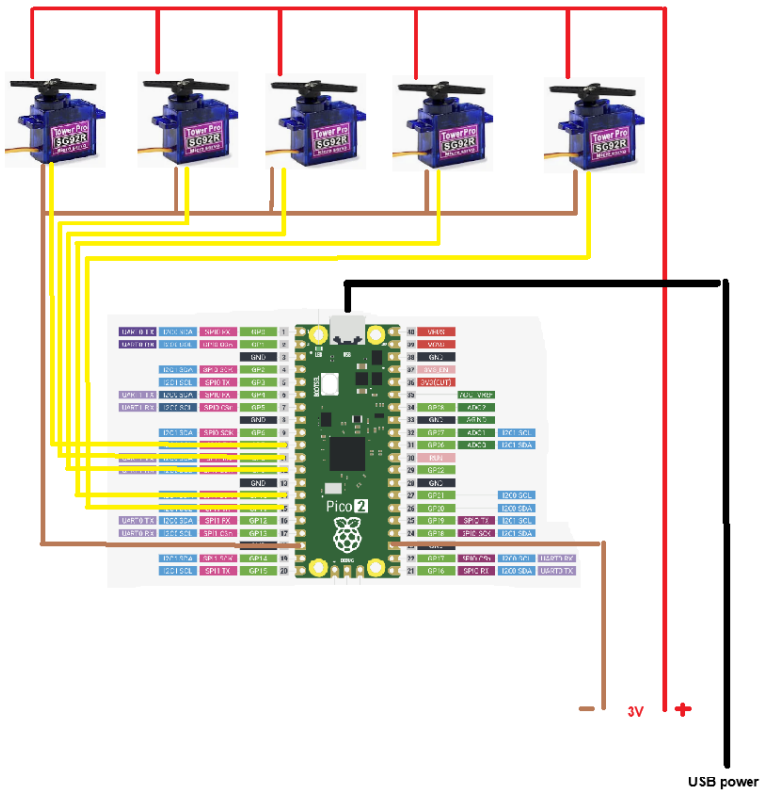
- Pico W receives angle values via MQTT
- Each value is clamped (e.g., thumb to 0–120)
- PWM updates servos → robotic hand moves

E. FEEDBACK / VALUE:

- Visual feedback on touchscreen & Ubidots dashboard
- Physical hand movement is real-time and wireless via clous
- Value = intuitive, portable control of robotic hand from anywhere

---

Simple wiring diagram

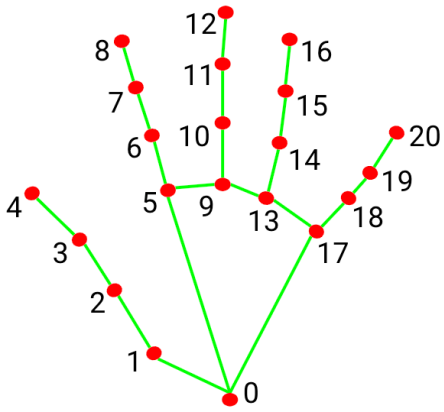


Electronically, both systems are nearly identically wired, with differences being only in pins used for PWM. The servos use an external power source which was made by dissecting a USB cable and soldering a 3V wire and the Ground to connections and then plugging the cable into a phone charger.

Code & Algorithm

All the code used can be found on my GitHub repository: https://github.com/GCM-Novosel/IOT-project_CV_Robotic_Arms

The **Computer vision Controlled Hand** uses MediaPipe HandDetector AI model that outputs an array of live positions of detected hand landmarks in a format of pixel positions (X,Y). I used this to calculate the "openness" of each finger by calculating the length of the position of each fingertip with regards to a reference point.



Then I divided all 5 values with a reference length of a palm to get the same values whether the hand is close to the camera or far away from it.

After that I normalized the values into ranges between 0-180 with each servo having it's own specific range, not to pressure them over the possible physical angle. These values are then written to corresponding PWM pin to dictate the angle of the servo.

The **MQTT-Controlled Robotic Hand** operates based on finger positions received over the internet via the Ubidots MQTT broker. The control interface runs on an Orange Pi with a touchscreen and allows users to either toggle fingers individually or send predefined gestures, which are then translated into sets of servo angles.

Each finger is assigned to a specific topic on the Ubidots platform (e.g., /v1.6/devices/orangepi/index) and receives values in real time. The Orange Pi publishes angle values (between 0–180°) to each topic, which are then picked up by the connected microcontroller (Raspberry Pi Pico W).

Gestures such as "peace", "fist", or "rock_on" are internally mapped to servo angles for each finger. Here is an example of how this mapping is implemented in code on the Orange Pi:

```
POSES = {
    "peace":    [0, 0, 180, 180, 120],
    "rock_on":  [0, 180, 180, 0, 120],
    "fist":     [180, 180, 180, 180, 120],
    ...
}
```

Each gesture corresponds to a list of angles for the fingers [index, middle, ring, pinkie, thumb]. When a user selects a gesture on the UI, these angles are sent over MQTT, one for each finger.

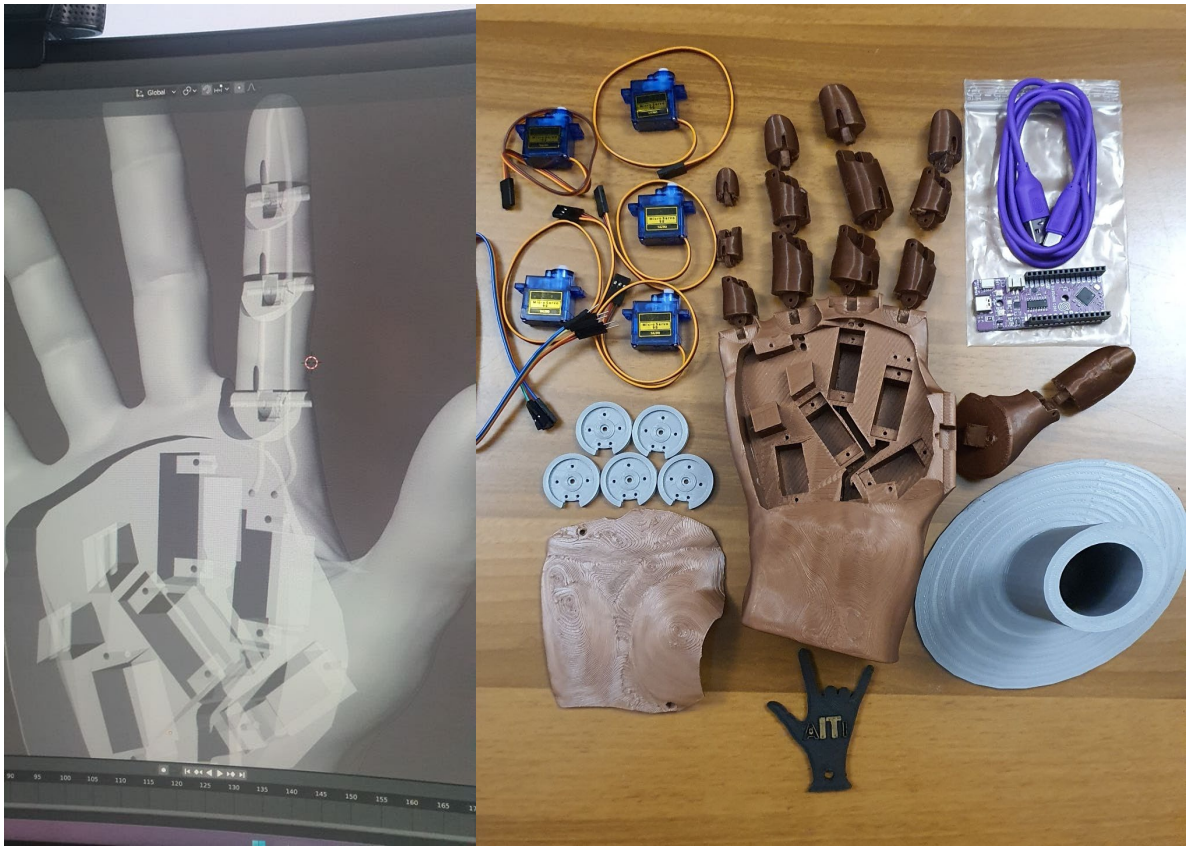
On the receiving end, the Raspberry Pi Pico W subscribes to these MQTT topics. Upon receiving a new value, it decodes the payload, clamps the angle to a safe servo-specific range (e.g., 0–120° for the thumb), and sets the PWM signal on the corresponding pin. The duty cycle is dynamically calculated, and an onboard LED blinks briefly to confirm successful updates.

This setup allows flexible, real-time wireless control, whether adjusting individual fingers or sending complex gestures in one touch.

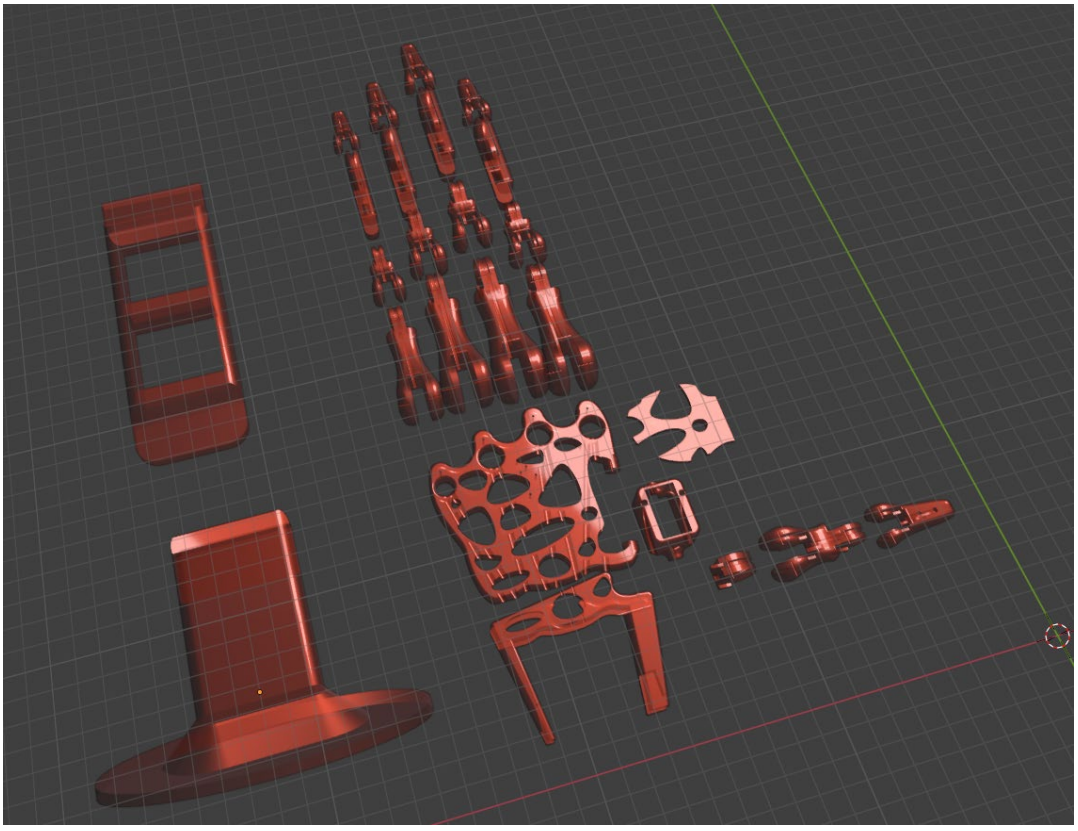
3D Models, 3D printing and mechanical aspect

I designed the first one completely by myself in Blender, and for the second one I have found a great model of a skeletal hand that I modified to my needs.

I printed everything with my own 3D printer and used whatever I could think of to be useful, like fishing wire, needles, jewelry etc.



All the models were designed with keeping attention to the mechanics, angles, routes of the fishing wires, at the finished product's smoothness is a result of a lot of testing, simulating and iterating. Still, there is room for more improvement.



Evaluation

Item	Max	My score	Motivation
Original concept	2	2	The idea of building two robotic hands, one controlled by computer vision and the other over MQTT with a custom UI demonstrates creativity, relevance, and strong integration of modern IoT concepts.
Interfaces	8	8	GPIO, PWM used extensively for servo control. SPI + LCD attempted integration (Nokia 5110, with wiring and code prepared) (Code present in repo)
MQTT via Pico	4	4	The MQTT implementation is solid: Topic-based finger control with real-time feedback, error handling, and blinking LED indicators. Full use of Ubidots cloud integration and topic subscription handling.
Dashboard	2	2	Ubidots dashboard used to send and visualize finger data in real-time. Clear communication between the Orange Pi and the Pico via MQTT cloud broker.
Extra: AI, ...	2	2	The first robotic hand leverages AI-based computer vision (MediaPipe) for hand detection. Finger openness is calculated dynamically using landmarks and scaled to servo angles. This adds a real AI element to the project.
Nice demo / video / report	2	2	The YouTube video demonstrates clear functionality, smooth control, and both interfaces in action. The GitHub repo is well-structured and documented, and this self-assessment/report clearly outlines the project scope and success.

Total 20/20

By:
Tomislav Novosel