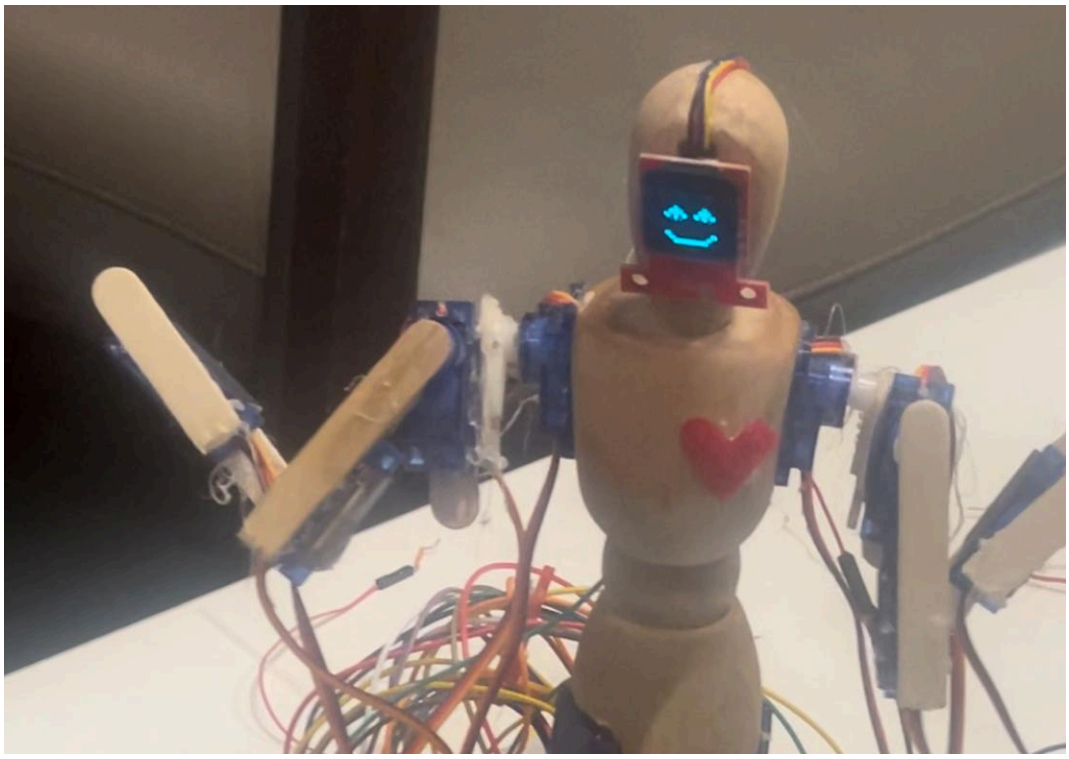
University of Southern California
On Making Smart Devices
Spring 2024

FINAL PROJECT

Developer Documentation



Magen Mozeh

mozeh@usc.edu

TABLE OF CONTENTS

Author's Notes

This guide provides an overview of the robot's functionalities and components, enabling future modifications and enhancements. While the videos showcasing real-time operation may experience occasional lag, the face updates revert to idle animations smoothly. It is advisable to secure the limbs more firmly to enhance product durability. Overall, the functionality is robust, with angles being accurately calculated and adjusted accordingly.

Overview

The "Development of a Smart Robot for Children" project focuses on creating an interactive robot tailored specifically for children, equipped to mimic human-like movements and express various moods. Using servos controlled via the Particle platform, the robot can perform actions such as waving and kicking and mimicking poses, while its mood simulation is managed through a MicroOLED display and sensors, allowing it to exhibit emotions like happiness, sadness, anger, and surprise. These features make the robot dynamic and responsive, enhancing engagement with young users.

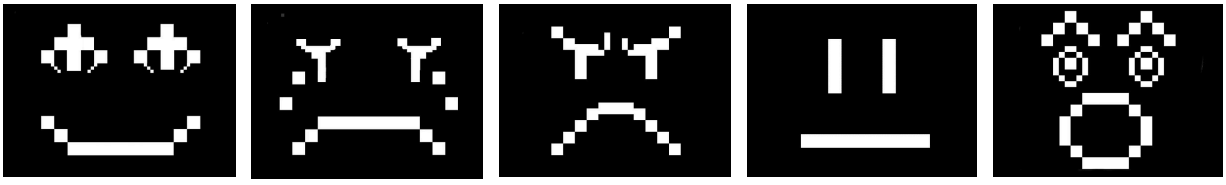
This technology offers a compelling alternative to the typical screen-centric interaction that often results in children becoming detached while using their phones. By leveraging facial and body detection for device control, it encourages children to be more physically active and aware of their surroundings, promoting a more embodied form of engagement. This approach moves away from passive screen tapping and swiping, promoting a more immersive experience. Such interactions can help in developing better motor skills, spatial awareness, and a deeper connection to the physical world, contrasting sharply with the static, isolated nature of conventional smartphone use. Simply walking in front of the device reacts to you!

The user can switch between

a) Body Mimicking: This feature allows the robot to replicate human-like movements, utilizing servos controlled via the Particle platform. It enables the robot to perform a variety of actions such as waving, walking, or dancing, mimicking the user's gestures or predefined sequences. This interaction helps in making the robot appear more lifelike and engaging, especially for children.

b) Facial Recognition and Emotion Display: The robot is equipped with facial recognition technology, which enables it to identify and react to different users. Coupled with emotion detection, the robot can analyze the user's facial expressions to interpret emotions and respond appropriately. This is displayed through the MicroOLED screen, which can show expressions of happiness, sadness, anger, and surprise, enhancing the emotional interaction between the robot and the user.

I converted these expressions into graphical bitmaps for display on the OLED screen:



HAPPY **SADNESS** **ANGER** **NEUTRAL** **SHOCKED**

DISGUSTED **FEAR**

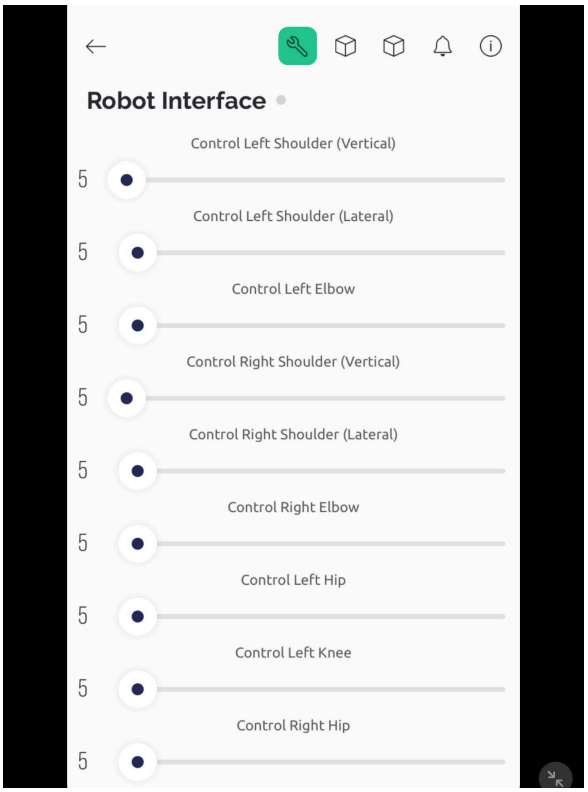
Blynk App and Using the Device

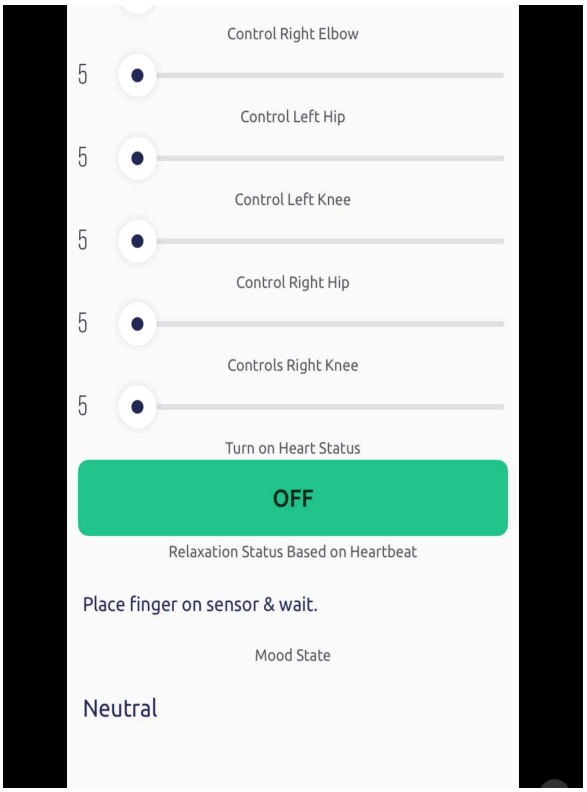
Open the command palette and cloud flash the device after configuring it to your device. You can open the facial functionality via the python script and body detection.

In terms of an interface

1. Sliders:

The interface allows for manual control of the robot's joints. This feature is particularly useful for users who prefer setting specific poses manually, similar to adjusting an action figure. It offers an engaging way for children to interact with the robot by choosing poses that mimic their favorite characters or action figures. This can be done in real time or preset for convenience. This is done via event driven programming (arg





Utilizing the Heart Rate Sensor:

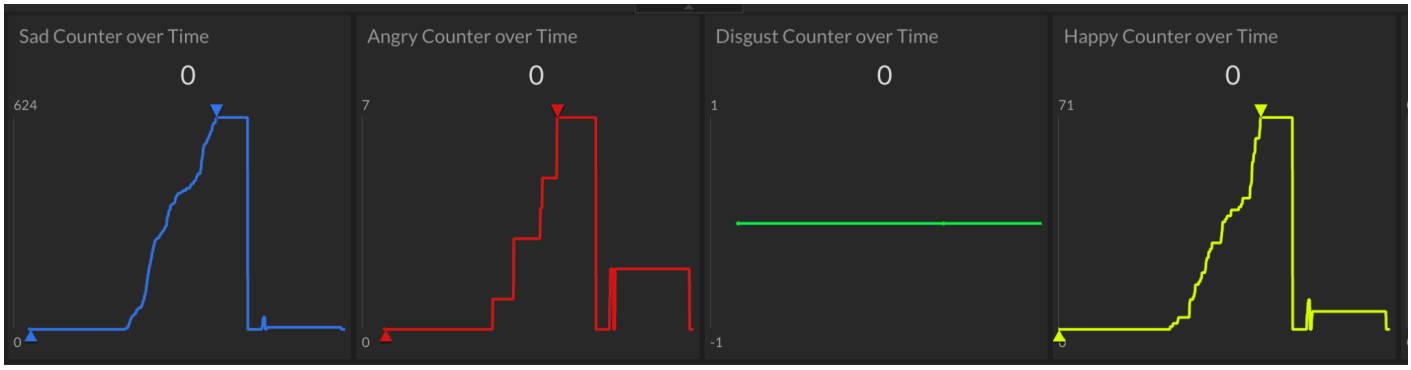
Place your finger on the sensor located where the robot's heart is depicted, and wait for the reading. This process activates the heart rate measurement.

Interactive Mood Display with Camera Integration:

Activate the robot's camera to enable the mimicry feature. As the robot mirrors your movements, it will also display corresponding moods on the app, enhancing the interactive experience.

Initial State Variables- Getting Data Information

In my project, I implemented a counter to track each time a specific emotion is triggered and displayed on the OLED screen. This allows the system to log how frequently different emotions, such as happiness, sadness, anger, and surprise, are recognized and shown. Over time, this data provides valuable insights into which emotions are most commonly displayed, helping to analyze the system's interaction patterns and emotional responses.



Additionally, I've identified the main facial expressions commonly made by individuals and incorporated a counter for each. This setup counts and records the frequency of each detected expression whenever the facial recognition feature is utilized, providing detailed analytics on interaction patterns.

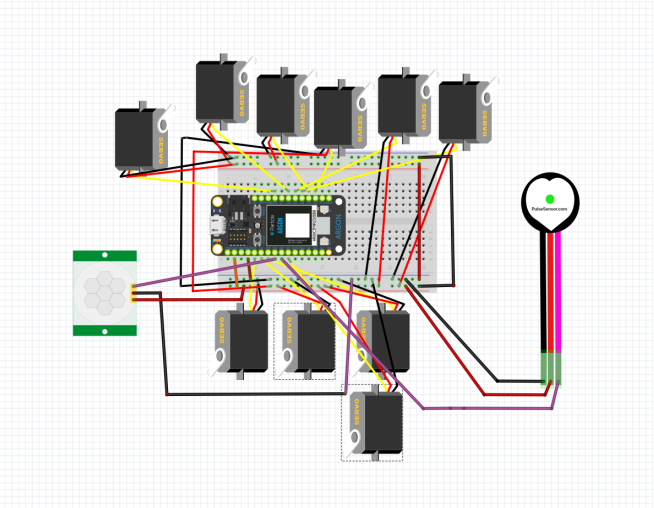
Implications that the data imposes:

1. Emotional Engagement: Frequent changes in facial expressions can indicate high levels of engagement and interest in the robot. Conversely, a lack of variation might suggest that the robot is not effectively capturing the child's attention or needs adjustment in its interactive features.

2. Longitudinal data can highlight trends in behavior, such as increased confidence or decreased anxiety, which are useful for caregivers and educators to monitor the overall well-being and development of children from the robot.

3. Predictive Behaviors: With enough data, patterns can begin to predict how a child might react in certain scenarios, allowing for proactive adjustments to the robot's behavior that preemptively cater to the child’s needs.

Fritzing Diagram and Functionality



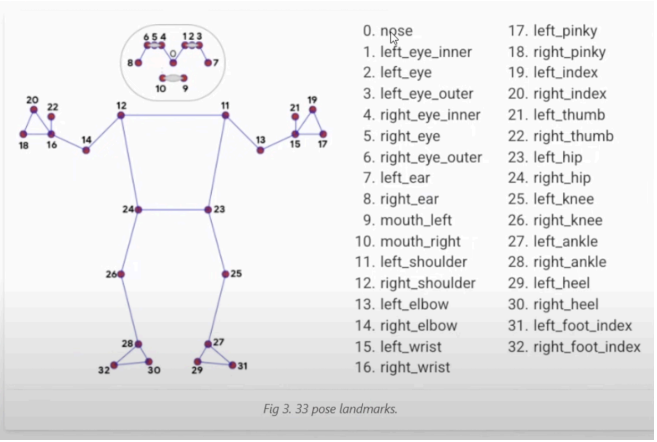


Fig 3. 33 pose landmarks.

10 servos - Joint Control

The robot features arms that can move both vertically and laterally, complemented by articulated elbow joints, while its hips and knees are designed to bend, enabling it to mimic human movements accurately in real time. For facial emotion detection, I employed the Deepface library, which utilizes a comprehensive database of images linked to various emotions, enhancing the robot's ability to recognize and react to users' facial expressions accurately. Additionally, I integrated OpenCV and MediaPipe, which provide precise landmarks for the joints, allowing for real-time motion tracking and analysis.

In total, I employed three servos per arm and two per leg. This setup allows the arms to execute complex movements like inward muscle flexing and the legs to perform actions such as kicking, enabling realistic, real-time flexing and movement.

OLED Screen - Facial

The OLED screen updates to reflect mood states detected by the Deepface library, displaying different expressions based on the emotions identified.

Open PIR (Passive Infrared Sensor) - Detects human nearby

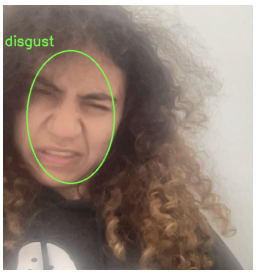
Humans emit a different level of infrared radiation compared to the surrounding environment. When a device reacts to a child's presence by displaying a face or character, it can capture their attention and stimulate curiosity. This interaction can be used as a teaching tool, where children learn about cause and effect ("When I move, the robot sees me and reacts").

Heart Monitor

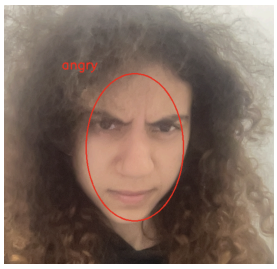
Children can utilize the heart rate monitor to assess their current state of relaxation. The robot will interpret the readings and suggest appropriate activities. These recommendations will appear on the app once the measurement is triggered.

Device Instructions

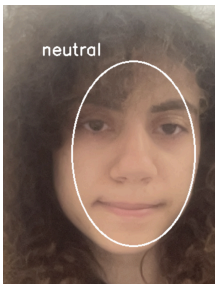
If not using the aforementioned interference simply just stand in front of the camera for real-time!



Disgust

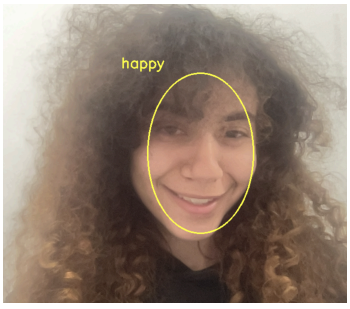


Anger

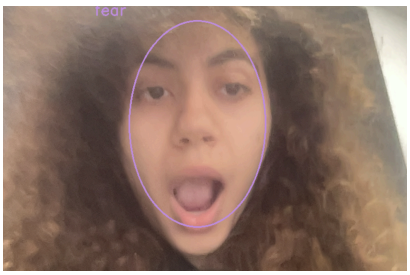


Neutral

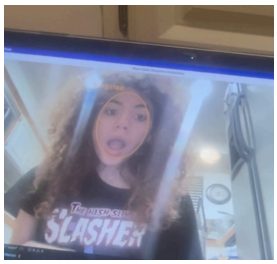
Sad



Happy

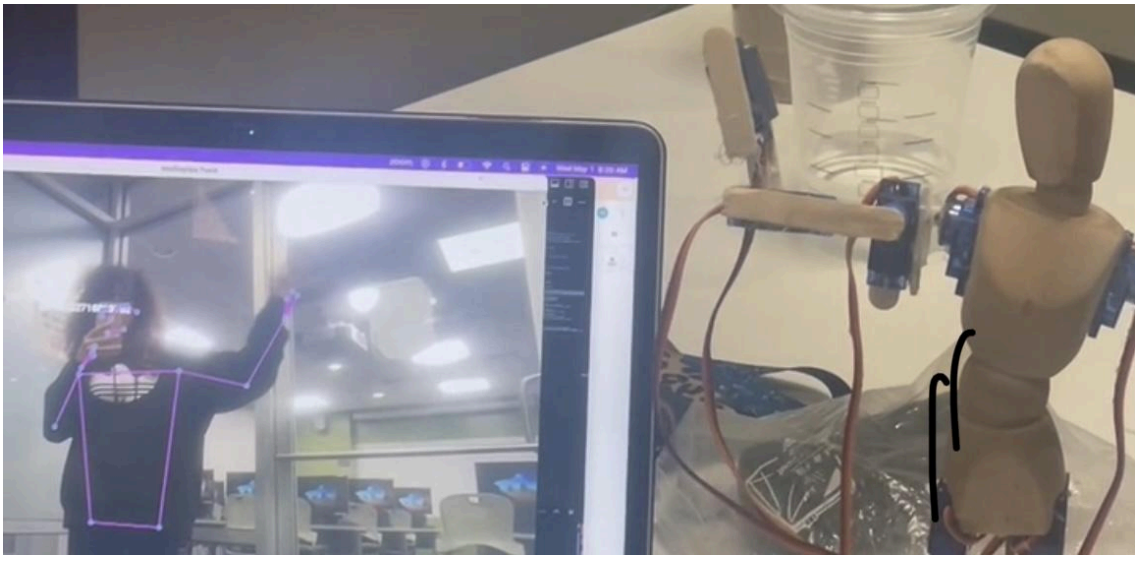


Fear



Surprise

Press ‘Q’ to quit any one of the facial or body detecting interfaces.



Heart Sensor

To start, press on the sensor to trigger the heart eyes animation, indicating readiness. The robot displays an "in love" animation by alternating between two bitmaps, which also serves to teach children about expressions of affection.

Depending on the heart rate measured, the robot will provide one of the following messages to the Blynk app:

- Below 60 BPM: "Very calm, ideal for rest."
- 60 to 70 BPM: "Calm, good for learning."
- 71 to 85 BPM: "Normal activity, no stress."
- 86 to 100 BPM: "Mildly elevated, slight excitement."
- 101 to 115 BPM: "Elevated, may be stressed or active."
- Above 115 BPM or unclear readings: "Please place your finger on the sensor and wait."

Summary of Code

BodyDetection.py:

1. Libraries and Initialization:
    - OpenCV (cv2): Used for handling video capture and display.
    - MediaPipe (mp): Utilized for its pose estimation capabilities. This library provides highly accurate pose landmarks.
    - Numpy (np): Supports numerical operations on arrays, crucial for calculating angles between landmarks.
2. MediaPipe Pose Setup:
    - An instance of MediaPipe Pose is initialized with minimum detection and tracking confidence levels set, preparing it for real-time pose detection.
3. Video Capture:
    - The script captures video frames from the webcam (cap = cv2.VideoCapture(0)). Each frame is processed to detect human poses.
4. Frame Processing:
    - Each frame is converted from BGR to RGB color space because MediaPipe requires RGB inputs.
    - Pose landmarks are detected in each frame.
5. Landmark Detection and Angle Calculation:
    - Specific landmarks (like ankles, knees, hips, shoulders, elbows, and wrists) are identified.

- Angles at various joints (like elbows, shoulders, and knees) are calculated using a helper function calculate_angle(), which computes angles based on the positions of three points.

6. Servo Angle Communication:
   - Calculated angles are sent to a remote server via the Particle Cloud API, controlling servos that presumably mimic the user's movements. This involves formatting the angle data and making an HTTP POST request.
7. Visualization and Display:
   - The pose landmarks and their connections are drawn on the image for visualization.
   - The modified image is displayed in a window, allowing the user to see the pose estimation in real-time.
8. Exit Condition:
   - The script continuously captures and processes video until the 'q' key is pressed, at which point it terminates the video capture and closes any GUI windows.

For the angle calculation:

```
radians = np.arctan2(c[1]-b[1], c[0]-b[0]) - np.arctan2(a[1]-b[1], a[0]-b[0])
angle = np.abs(radians*180.0/np.pi)
if angle > 180.0:
    angle = 360 - angle
```

This snippet calculates the angle between three points, which are used to determine the angles of joints. The result is crucial for accurately mimicking human movements with servos

For Sending Servo Angles to the Particle API
response = requests.post(PARTICLE_API_URL, data=data)

emotion.py:

1. Emotion Detection:
   - For each detected face, the region of interest (ROI) is analyzed using the DeepFace library to determine the dominant emotion.
   - The detected emotion is then sent to a remote API endpoint using a POST request, which could be used to trigger reactions or log data.
2. Visual Feedback:
   - The script uses an ellipse to visually highlight detected faces in the video frame. The ellipse and the emotion label are colored based on the detected emotion, using predefined color mappings.
   - Emotions and corresponding colors include yellow for happiness, blue for sadness, red for anger, and other colors for additional emotions like surprise, fear, disgust, and neutral.

```
faces = face_cascade.detectMultiScale(gray_frame, scaleFactor=1.1, minNeighbors=5,
minSize=(30, 30))
```

result = DeepFace.analyze(face_roi, actions=['emotion'], enforce_detection=False)

- This part detects faces in the frame and then analyzes the emotional expression of each detected face.

response = requests.post(PARTICLE_API_URL, data=data)

- This sends the detected emotion as a string to a remote server, potentially to trigger actions or log the emotion for data analysis.

cv2.ellipse(frame, center, axes, 0, 0, 360, color, 2)

cv2.putText(frame, emotion, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, color, 2)

.cpp particle file:

- This snippet draws an ellipse around each detected face and annotates it with the detected emotion.
- Main Loop (loop() Function):
    - Handles Blynk server communication to maintain IoT connectivity.
    - Processes inputs from a PIR sensor to detect presence and trigger specific animations or alerts on the OLED.
    - Regularly updates servo positions based on target angles which can be modified via the Blynk app, allowing dynamic control of the robot's posture.
    - Periodically checks and updates the display based on the robot's mood state, managed through custom animations and bitmaps.
    - Measures and updates heart rate information at set intervals, providing feedback through the Blynk app.
- Mood and Animation Management:
    - Functions are provided to display different moods using custom bitmaps, enhancing interaction by changing the robot’s expressions according to its emotional state.
    - Idle animations and love animations are used to provide visual feedback and engage with the user, potentially in educational or therapeutic contexts.
- Remote Control via Blynk:
    - Exposes functions to Blynk for remote procedure calls, allowing users to change servo angles (thereby adjusting the robot's pose) and set the robot’s mood through the app.
    - Heart rate data and mood states are sent to the Blynk app, providing a real-time interface for monitoring and interaction.
- Servo Control:
    - Detailed servo control is embedded within the loop, adjusting the robot’s limbs for predefined poses or responses to user interactions.

2 key functions

1. Particle.function("displayEmotion", displayEmotion); - This line exposes the displayEmotion function to the Particle Cloud. It allows remote users to change the robot's displayed emotion by sending commands like "happy" or "sad" from the Particle Cloud interface.
2. Particle.function("setServoAngles", setServoAngles); - This line exposes the setServoAngles function to the Particle Cloud, enabling remote adjustment of the robot’s servo angles. Users can send specific angle configurations as a comma-separated string to pose the robot differently.

3. To make both files talk to each other I created an access key and token. And sent over through a JSON by importing requests.

Photos