ARGUS – VIOLET

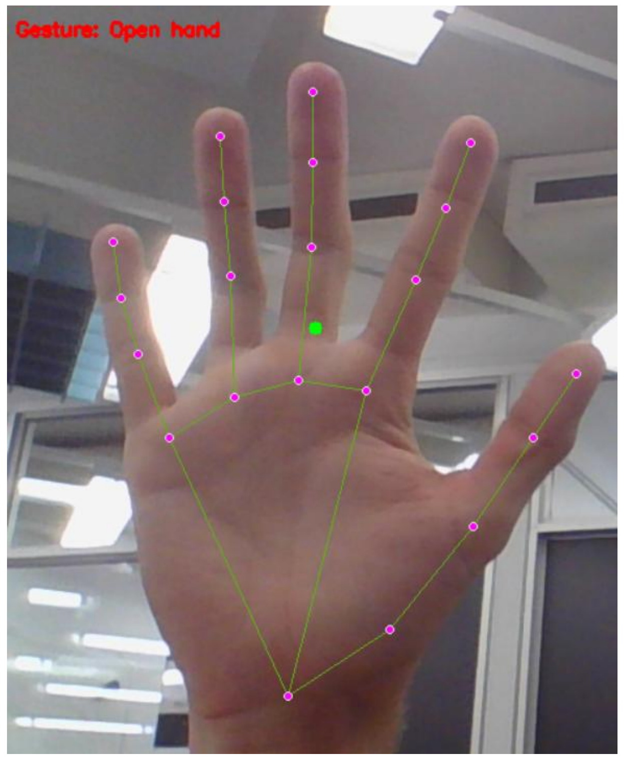
Domonic Richardson 46964584, Archie Wills 47440188, Liam Ryan 4749048

INTRODUCTION

Argus-Violet is a smart camera tracking system utilising machine learning, built around an ESP32-EYE mounted on a two-axis tilt and swivel servo, a PC interface, and a Keras sequential neural network model. The aim of Argus-Violet is to use a ML model to predict hand gestures in the frame and use these gestures to control the movement of the ESP32-EYE mounted on pan and tilt servo bracket.

Gestures

- Open palm – resets servos to centre
- Closed fist, thumb out – Moves to centre fist in frame
- Index Finger up – Moves up
- Index and middle figner up – Moves down
- Index and thumb L shape– Moves Left
- Three fingers up – Moves Right



SYSTEM OVERVIEW

ESP32-S3-EYE

The ESP32-S3-EYE running zephyr, uses zephyr video capture library to capture frames. Then using the zephyr networking library, frames are streamed to the Bridge node via UDP. The captured frames are also displayed on the lcd display on the EYE.

ESP32-DevKitC – Servo Node

The servo node is responsible for driving the servos that the EYE is mounted on. The servo node waits for commands sent from the bridge node via UDP.

ESP32-DevKitC – Bridge Node

The Bridge node listens for frame packets from the EYE via UDP. The bridge node receives the frame data and passes via UART to the PC interface. Additionally, the bridge node passes servo adjustment commands from the PC interface to the servo node.

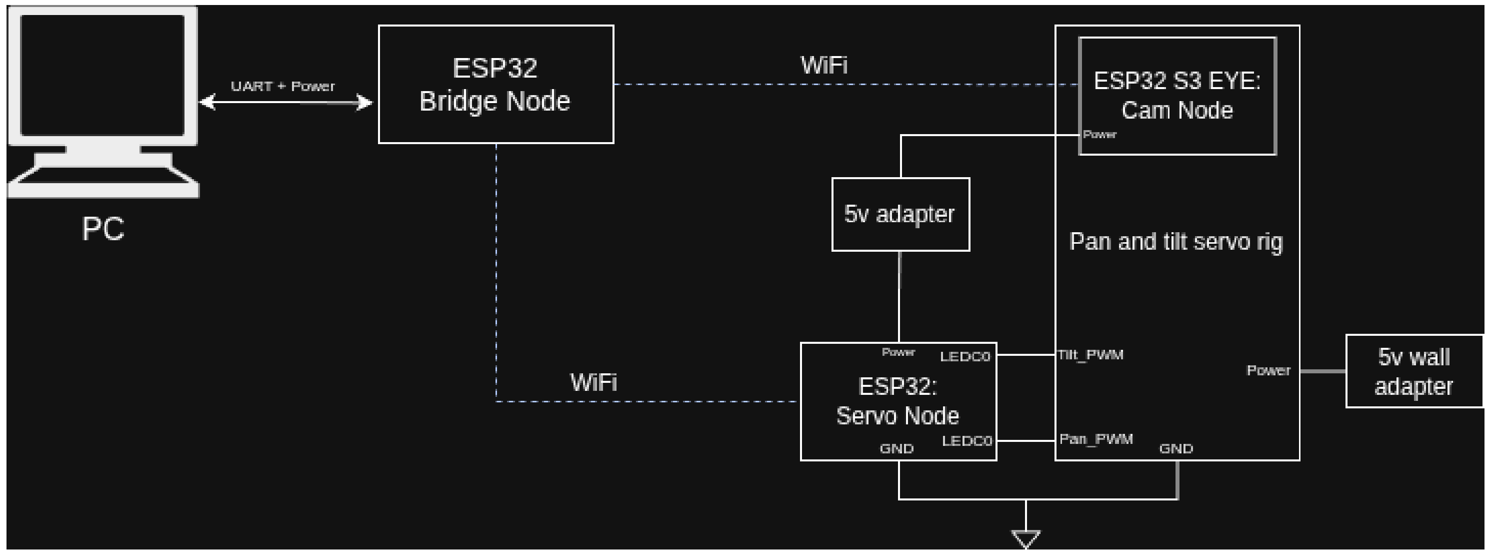
PC Python Interface

The PC interface reads frames from UART and applies some preprocessing to get hand landmark locations using MediaPipe library. Landmarks are passed to the Keras model, and a predicted gesture is returned and a corresponding command is sent to the servo node

Gesture Detection Model

The Model is a Keras sequential model. Keras is a high-level neural network API built on top of Theano and TensorFlow. The Model is input relative hand landmark locations, captured using Google MediaPipe hand landmark detection library.

BLOCK DIAGRAM



FINDINGS

KPI's

1. Frame rate on PC interface ≥ 0.5 FPS

Failed, Need to compress frames to reduce size of data being sent.

2. Successfully classify gestures > 80%

Success. ML model accuracy = 0.9751

3. Time between gesture in frame and repositioning of camera < 5 seconds

Limited by framerate, when PC receives frame, time to servo repositioning is < 1 second.

4. Average offset actual frame centre and the position of the hand in the frame after repositioning ≤ 15% frame width/height

Success, moves hand into 30*30-pixel centre box

5. ≥ 95% of servo commands sent from the PC are received and executed correctly by the ESP32-CAM.

Success, UDP commands rarely fail.

Challenges

The main challenge of the project was sending video frames via Wi-Fi. Our approach is to split the RGB565 byte array into chunks to send via UDP and then reassemble on the PC software. This method is slow and still requires sending the full 11kB frame. To speed up the frame rate some compression method needs to be utilized, however there is limited support on Zephyr for video compression.

REFERENCES AND ACKNOWLEDGEMENTS

Google MediaPipe Hand Landmark detection - https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker

Zephyr Docs – https://docs.zephyrproject.org/latest/index.html

TensorFlow Keras – https://www.tensorflow.org/guide/keras