INTRODUCTION

TO

ESP32 ROBOTICS SYSTEMS

By

Andrew R. Sass

TABLE OF CONTENTS

SUMMARY

The proliferation of the ESP32 series of System-On-Chip (SOC) microcontrollers makes possible the development of web-based Robotics Systems allowing both operator and autonomous operation.

The Systems described here are complete front-ends and include an Arduino Mega as the engine able to host actuators and sensors for robot operation. The Systems are not limited to the Mega which can be replaced by a Raspberry Pi or an even higher performance processor. The low cost of these systems makes them ideal for a broad range of robot applications, both consumer and industrial.

This paper is organized into 2 parts (see Table of Contents):

- **PART 1**: general description of the systems
- **PART 2**: specific program descriptions and the programs

It is important to read Part 1 first for each system. Part 2 is the next layer of the “onion”, repeating only necessary bits of information in Part 1 to “bridge” Part 1 to Part 2. An annotated listing of the programs is contained in Part 2 as well as a GitHub link to the programs for easier download access (Page 43).

HARDWARE

The hardware used in this project is available on AMAZON as follows. Prices quoted are as listed during Summer 2021. Items are available elsewhere at potentially lower prices but the following list is a starting point.

ESP32	TTGO T-Journal ESP32 Camera Module Development Board OV2640 Camera WiFi 3dbi Antenna OLED Display	$16.85+delivery
Arduino PRO Mini	HiLetGo PRO Mini Atmega 328-AU 5V/16MHz 3pcs	$13.49
OLED Display	HiLetGo 1.3” SPI 128X64 SSH1106 OLED LCD Display	$8.99
Arduino MEGA	ELEGOO MEGA 2560 R3 Board Atmega 2560 + USB Cable	$15.99

The ESP32 used here (TTGO T-journal) was chosen for two reasons:

1. The I2C interface—ESSENTIAL
2. The installed external antenna—CONVENIENT. The word convenient is not used lightly as all ESPs come with an on-board printed circuit board antenna which can be disabled and an external antenna can be attached. However, the disabling process can be challenging and potentially destructive to the ESP32. Users

can certainly proceed with the on-board antenna with lower *WiFi* range capabilities.

For the purists who can't resist the inherent challenge, an alternate ESP32 with an I2C interface can be purchased. An external antenna is available at Amazon. For example *Bingfu* Dual Band *WiFi* 2.4 GHz, 5 GHz,5.8GHz 6dbi RP-SMA (2pcs $8.99) **However, YOU HAVE BEEN WARNED!!!**

PART 1

GENERAL DESCRIPTION OF THE SYSTEMS

ESP32

INTRODUCTION

ESP32 is a series of system-on-a-chip microcontrollers with an integrated Wi-Fi 802.11 b/g/n capability, including a built-in antenna (or capability for an external antenna. The microcontroller has ADC, DAC, SPI, I2C, I2S, and other capabilities including **models with an on-board camera which is key to the robot applications of interest here**.

Most impressive is its ability to operate within the Arduino IDE thereby opening it up to the Arduino programming community. Additionally, it can be programmed within VS Code, Platform IO. Excellent descriptions of applying the ESP32 have been published by Rui Santos (Random Nerd Tutorials); reference links are given below:

- https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/
- https://randomnerdtutorials.com/vs-code-platformio-ide-esp32-esp8266-arduino/

In addition, randomnerdtutorials.com contains a treasure trove of well-written tutorial information on ESP32 and ESP8266 projects including several e-Books on specific areas of interest.

BACKGROUND

The author first explored the use of the ESP32 as a component for an Autonomous Robot in a project that was featured in an RNT tutorial.

- ESP32-CAM Web Server with OpenCV.js: Color Detection and Tracking | Random Nerd Tutorials

The project explored the extension of OpenCV to OpenCV.js (JavaScript); thereby making possible the introduction of OpenCV to ESP32 CAMERA WEB SERVER applications.

The project introduced the subject of OpenCV.js, and introductory tools of OpenCV to the ESP32 CAMERA WEB SERVER environment. It was by no means an exhaustive treatment of all that OpenCV can offer to ESP CAMERA WEB SERVERS. It explored a few of the basic concepts of OpenCV in a representative ESP CAMERA WEB SERVER program which involved color tracking of a moving object.

That project formed the starting point for the current, and more ambitious work described here; Robotics Systems featuring both Operator controlled, henceforth also referred to as **Teleop**, and **Autonomous** operation.

For the sake of completeness, and for the convenience of the reader, the material of that earlier work is included in this paper.

SERVER: STATION AP VS SOFTAP

The code in this paper focuses on using Station Access Point for both the ESP32 Browser Server and Python Streaming Server. However, as will be seen, a simple “define softAP” statement can be “commented in” to switch to the softAP mode for each Server. In the softAP mode, the Internet is not connected and the Client is accessing the ESP32 Server only via the Server IP address and Name.

An excellent reference for softAP is the following:

- ESP32 Access Point (AP) for Web Server | Random Nerd Tutorials

However, the reader is cautioned that for the case of the Browser Client, the line of code that accesses OpenCV.js via Internet must be “commented out” and the line accessing OpenCV.js locally must be “commented in”. Furthermore, to access OpenCV.js locally, the Chrome security feature must be considered or else it will throw a security exception. The reader will find a method to avoid this problem in the “Living with Chrome Security” chapter.

There are no such issues for the Python Streaming System Client since OpenCV is not located in the ESP32. It is resident in the laptop running the Client program.

The decision to use one or the other of Station AP vs softAP is left to the reader.

SERVER: BROWSER CLIENT (BROWSER STREAMING SYSTEM)

The ESP32 can act as a server for a browser client and some models include a camera that allows the client to view still or video pictures in the browser. HTML, JavaScript, AJAX, and other browser languages take advantage of the extensive capabilities of ESP32 and its camera. Those who have little or no experience with ESP32 CAMERA are referred to the excellent tutorials authored by Rui Santos in Random Nerd Tutorials.

As described in docs.opencv.org, “OpenCV.js is a JavaScript binding for a selected subset of OpenCV functions for the web platform”. OpenCV.js uses Emscripten, a LLVM-to-JavaScript compiler, to compile OpenCV functions for an API library that continues to grow.

It is available to anyone with only a modest background in HTML and JavaScript. Those with a background in Esp32 Camera applications have this background already.

SERVER: PYTHON CLIENT WINDOW (PYTHON STREAMING SYSTEM)

ESP32 can operate as a video streaming server for a Python client window. The Python client then can offer a more extensive library of OpenCV software than is currently available in OpenCV.js, because OpenCV is resident in the users’ laptop as opposed to the ESP32.

However, unlike the Browser Streaming System, data communication back to the server is not inherently available in the Python Streaming System. Since communication back to the server is essential for any complete robotic system, a novel method for this is presented in this work. It is, admittedly more complex, yet hopefully balanced by the advantages of the above-mentioned Python OpenCV library for those users not bound to a Browser application.

BROWSER STEAMING SYSTEM OR PYTHON STREAMING SYSTEM?

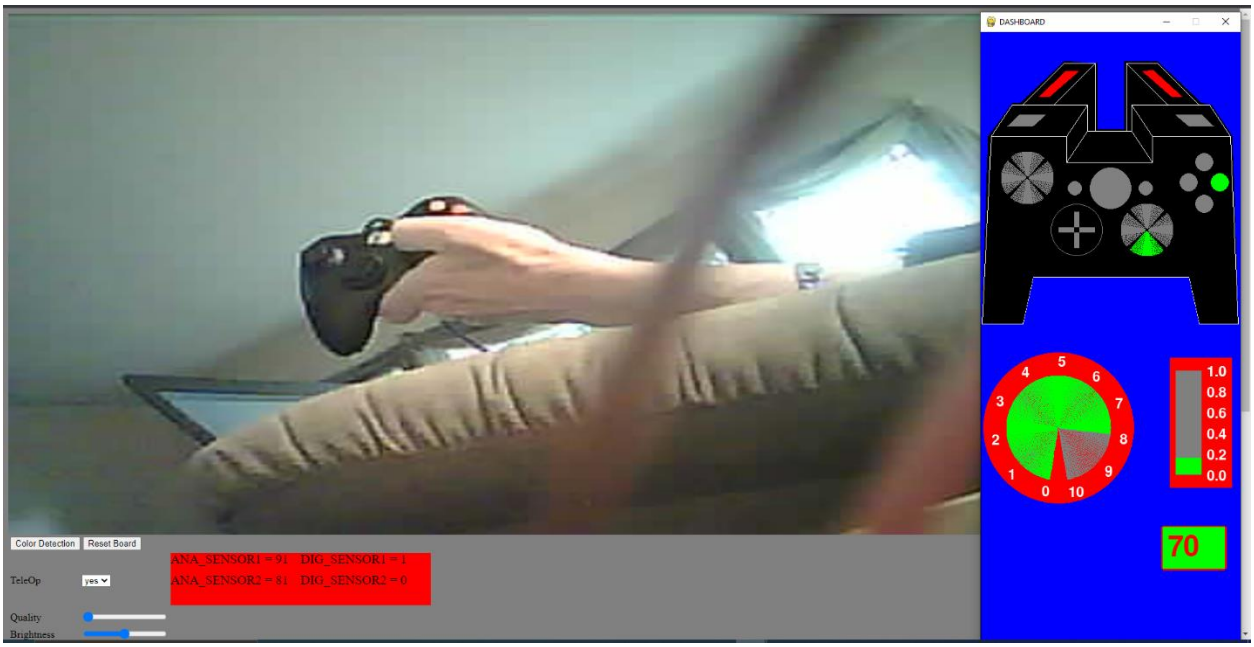
This paper will present a Robotic System for each alternative. Only the final user can pick the preferable alternative which depends on the user's environment.

ROBOT SYSTEMS

Following is a brief introductory pictorial description of the Browser Streaming System operation.

BROWSER STREAMING SYSTEM

The Browser Server (*ROBOT_BROWSER_SERVER.ino*, described later) publishes its IP address and is accessed by using a web browser. After pressing the **Color Track** button on the Chrome screen, the Client video screen is seen below in *Picture 1A*.

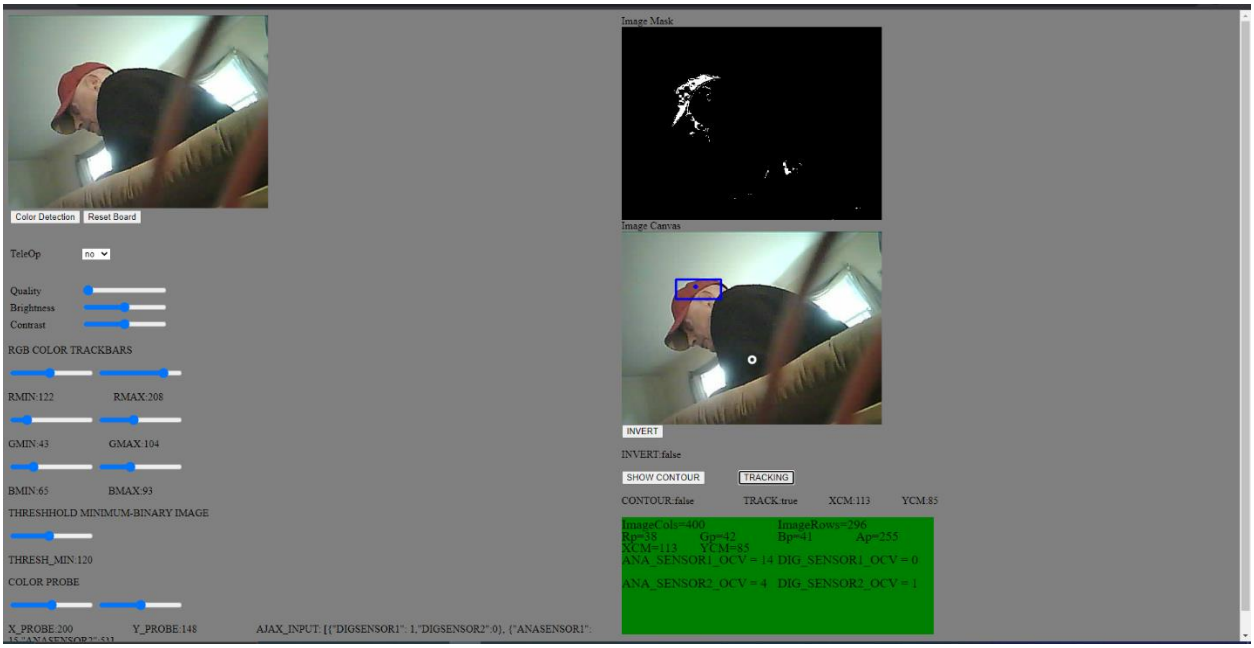


Picture 1A: browser streaming system client window, teleop with xbox controller GUI.

The red screen patch above shows sample sensor data transmitted by the ESP32 Server to the Client.

The XBOX Controller transmits Joystick, Button, Trigger, and Hat data to the ESP32. The ESP32 is the front end of the Robot and sends/receives data to the Robot's main processor. This mode, in which a human operator controls the robot, is called the **Teleop** Mode.

When the list box selector below the left side of the video is set to NO, the view in Picture 1B below is seen.



Picture 1B: browser streaming system client window, autonomous.

This is the Autonomous Mode in which the robot is under program control. In this mode, OpenCV (computer vision) is used to guide the robot. The top left view is the author wearing a red hat. The bottom right view shows a white dot approximately in the lower center screen. This dot is moveable and probes the position of the screen where the program can measure the RGB content of the picture. In this example,

the white dot probe was initially moved to the red hat where the measurement was made and then moved back to its initial position.

The RGB sliders were set to the red hat RGB measurement. The top right view shows the binary image corresponding to the color range set by the sliders. The hat is the largest portion of the image corresponding to the sliders' ranges. When the Tracking button is pressed, a blue contour rectangle surrounds the biggest single portion of white, identifying it as the target. The coordinates of the centroid of the blue rectangle are shown on the screen and transmitted to the ESP32 and then to the main processor of the robot for Autonomous guidance (Fig. 1 below). The Robot Browser Streaming System Diagram is pictured below in Fig.1.

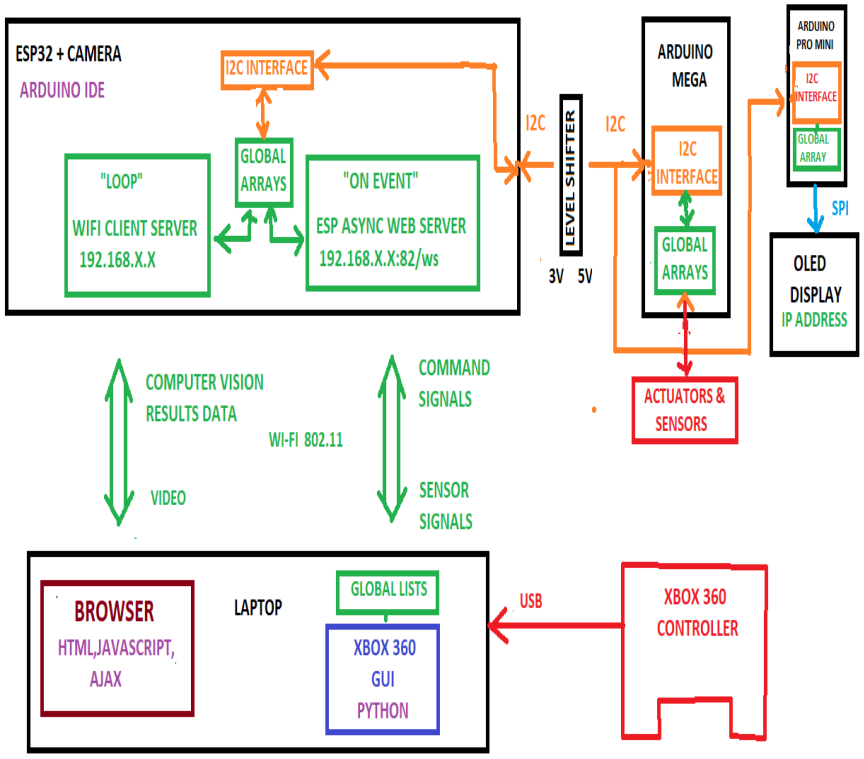


Fig. 1: browser streaming system diagram

The upper part of Fig. 1 is the hardware resident on the robot; the laptop and connected controller communicate with the ESP32 via WIFI.

(The author has used a CHROME BROWSER for the project described here.)

Starting from the lower right in FIG. 1, the XBOX 360 Controller is connected to a laptop that is running two programs: a Python program that presents the user with a GUI which is a pictorial representation of the Controller and shows which buttons, sliders, and hats on the Controller are being activated. The program features a Python List of 20 items representing the settings of the buttons, sliders, and hats.

This List is periodically transmitted to the ON-EVENT routine of the *AsyncWebServer* resident in the ESP32 and stored in a Global Array. The ON-EVENT routine immediately transmits data from another Global Array back to the GUI. Thus, two-way communication between GUI client and server is achieved and human operator control and feed-to-operator are made possible.

The reader can see that human-operator control can be traced from the Global Array receiving GUI data to an Arduino Mega via an I2C link; the Mega then controls all actuators and sensors by virtue of its pin capabilities which are superior to that of standardly available ESP32s. Also, sensor data from the Mega can be traced back via the I2C link to

the ESP32 Global Array responsible for data to be sent back to the Python GUI.

The ESP32 also hosts an ESP32 Server. A browser on the laptop, running as a client for the ESP32 Server, offers two modes of operation. One mode is a Teleop (human operator) mode which features a video feed from the Camera as well as a data feed from the ESP32 server in a JSON format. This data can be seen to originate back from the Mega sensors similar to the sensor data which propagated back to the GUI. System designers can decide which sensor data will be displayed in the Browser and which in the GUI.

The second mode of operation for the Bowser client is an “Autonomous” (machine-operated) mode in which the OpenCV.js portion of the Browser resident page executes a computer vision program whose results are sent back to the ESP32. The ESP32 then sends these results to the Megas’ actuators in a path similar to that seen in the Teleop mode, thereby achieving Autonomous operation.

The Autonomous data and the IP Address of the ESP32 are added to the 20-item array described for the Teleop mode as additional array items. It is then the entire array that is transmitted to the Mega.

The alert reader will note the Arduino Pro Mini in the diagram. For this paper, its sole task is to operate the OLED display which outputs the IP address of the ESP32 using an SPI link and the U8G2 protocol. Via this display, the user can avoid the requirement of needing to access the serial monitor of the Arduino IDE to determine the IP Address of the ESP32.

The use of the Mega and the Pro Mini in this paper can be considered somewhat symbolic as they can be replaced by other components which can perform the task described here.

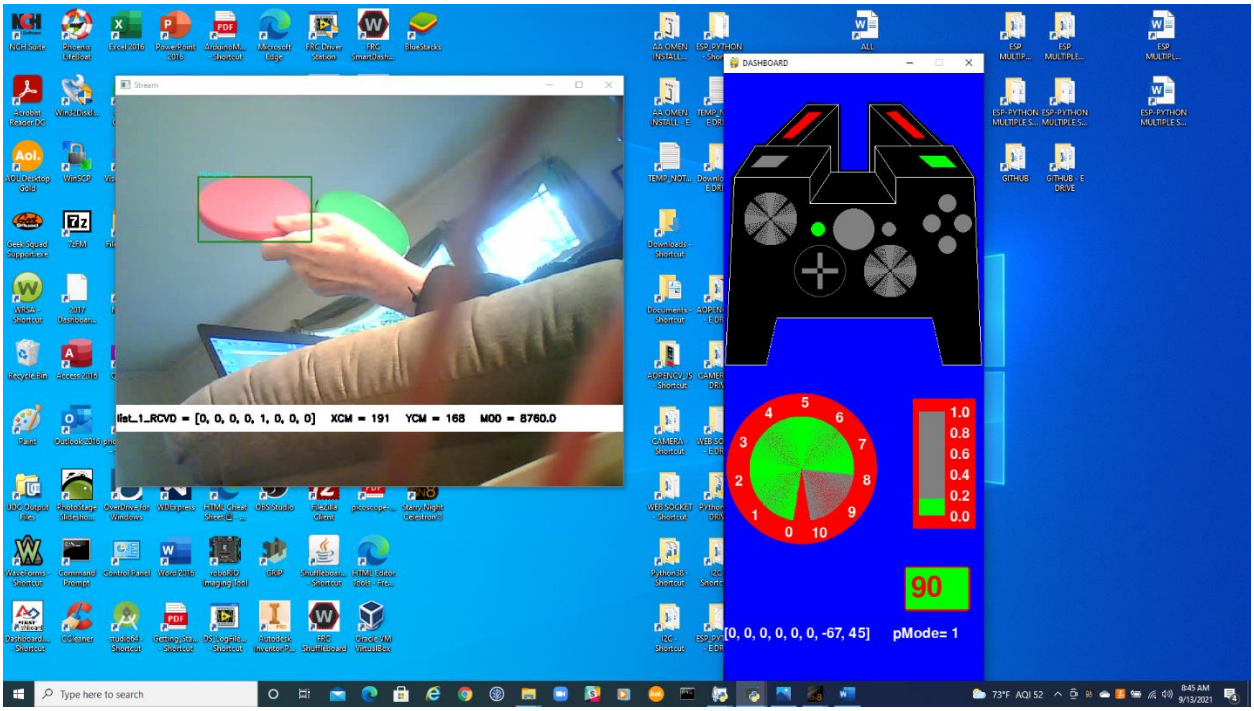
The example to be presented in this work is the classic OpenCV detection of the centroid of the colored object and the transmission of the centroids' Browser screen coordinates to the Mega. The Mega as part of a mobile robot can steer toward the target object based on the centroids' screen coordinates.

PYTHON STREAMING SYSTEM

The Streaming Server (*ROBOT_STREAMING_SERVER.ino*) to be described later, publishes its IP address immediately after download.

The *ROBOT_STREAMING_CLIENT.py* program (to be described later) uses this IP address and displays the python window shown at the left in Fig. 2 below.

The *XBOX_STREAMING_CONTROLLER.py* program, (also described later) also uses this IP address in Port 82/ws to display the Graphical User Interface(GUI) shown at the right in Picture 2 below.



Picture 2: The white stripe displaying data in the left window of the above figure and the white data at the bottom of the GUI are data exchanged by the two programs. In other words, the XBOX Controller

can control the program execution of the Client and the Client data can be used by the XBOX Controller and can be transmitted by the Controller to the ESP32 Server.

In the above figure, the XBOX Controller commands the Client program to identify the RED object. The Client program then transmits the centroid coordinated of the RED object to the XBOX Controller. If the human operator sets the Controller to Autonomous Mode, the centroid coordinates are transmitted to the Server and then to the main processor of the robot for guidance purposes.

The Python Streaming System Diagram is pictured below in Fig. 2.

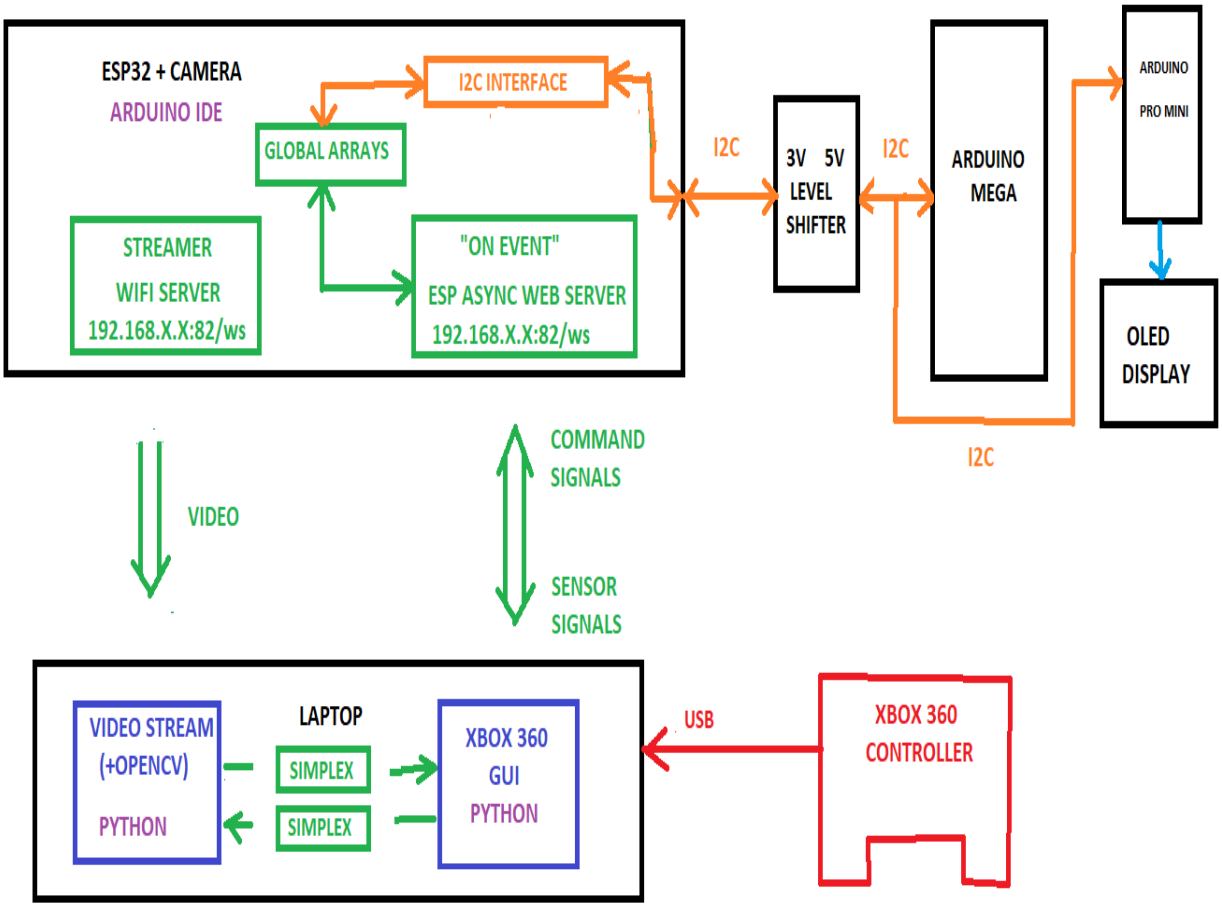


FIG. 2: The upper part of the Figure is the hardware resident on the robot; the laptop and connected controller communicate with the ESP32 via WIFI.

Again, starting from the lower right, the XBOX 360 Controller is connected to a laptop that is running two programs; a Python program that presents the user with a GUI which is a pictorial representation of the Controller and shows which buttons, sliders, and hats on the Controller are being activated.

The program, **similar but not identical to that in the Browser System**, features a Python List of 20 items representing the settings of the buttons, sliders, and hats. This List is periodically transmitted to the ON-EVENT routine of the *AsyncWebServer* and stored in a Global Array. The ON-EVENT routine immediately transmits data from another Global Array back to the GUI. Thus, two-way communication between GUI client and server is achieved and human operator control and feed-to-operator are made possible.

The reader can see that human-operator control can be traced from the Global Array receiving GUI data to an Arduino Mega via an I2C link; the Mega then controls all actuators and sensors by virtue of its pin capabilities which are superior to that of standardly available ESP32s. Also, sensor data from the Mega can be traced back via the I2C link to

the ESP32 Global Array responsible for data to be sent back to the Python GUI.

As discussed in the Browser System, the alert reader will note the Arduino Pro Mini in the diagram. For this paper, its sole task is to operate the OLED display which outputs the IP address of the ESP32 using an SPI link and the U8G2 protocol. **Via this display, the user can avoid the requirement of needing to access the serial monitor of the Arduino IDE to determine the IP Address**.

As in the Browser System, the use of the Mega and the Pro Mini in this paper can be considered somewhat symbolic as they can be replaced by other components which can perform the task described here.

Virtually simultaneously, a Python program (not a web browser) running as a client for the ESP32 Server offers two modes of operation: a video feed and an OpenCV capability. The video feed enables human operator, Teleop, and robot control while the OpenCV can be used for Machine, Autonomous, control.

It is here that the difference between the Python Streaming System and the Browser Streaming System can be seen. In the former system (the Python), there is no capability for OpenCV results to be directly returned to the ESP32; a critical requirement for Autonomous operation. It is therefore necessary to find another return path.

The alternate return path is accomplished via a simplex link between the Python streaming program and the Python XBOX Controller Program, allowing data flow from the former to the latter. During autonomous operation, operator data from the XBOX controller is not required for robot operation; this data is replaced in the Python XBOX controller program with the OpenCV results data. The OpenCV results data is then transmitted to the robot actuators and controllers via the same path as that for the Teleop signal data. As will be described in detail in the software section of this paper, the System is made aware of the Autonomous mode, as opposed to the Teleop mode via a single latched button press of the XBOX Controller.

The reader will note a second simplex link, enabling data flow from the Python XBOX Controller Program to the Python Streaming Program. This enables a novel alternative to the control of OpenCV functions by Trackbars. This alternative is the control of OpenCV functions by the buttons, hats, and joysticks of the XBOX Controller during the Autonomous setup.

As in the Browser System, the example presented in the software section is the classic OpenCV detection of the centroid of a colored object and the transmission of the centroids' screen coordinates to the Mega. In this case, the transmission is by way of the simplex link. The Mega, as described previously and based on the centroids' screen coordinates, can steer the host robot toward the target object.

XBOX CONTROLLER: USER INTERFACE

Both Robot Systems described above present a GUI (Graphical User Interface) to the user. The GUI is generated by a module, *Dashboard_26_module*, imported by the Xbox Controller program to be described in Part 2, the program section. This module is part of the development of this project and is described in the XBOX CONTROLLER GUI PROGRAM section along with a GUI picture and the module program listing.

The GUI presents a pictorial representation of an Xbox Controller on the Laptop screen. The pictorial, see Table of Contents-List of Figures and Pictures, shows a representation of the Controller and the various buttons, triggers, hats, and joysticks of the Controller.

The button color options are Gray or Green, the trigger colors are Red or Green, the Hat colors are Red or Green, and the joystick pictorial quadrants are partially (depending on joystick position) colored Red or Green. The buttons, triggers, hats, and joysticks colors are set by the Controller and the Controller program.

The state of the buttons is Gray to start, Green after a press and release, and back to Gray after another press and release, and so on. This will be referred to as a latched operation. Gray is 0 and Green is +1.

The triggers are Red to start, Green after a press and release, and back to Red after another press and release, and so on. This is also latched operation. Here, Red is 0 and Green is +1

The hats are Red for upper and left press and Green for down and right press where Red corresponds to -1 and Green to +1.

Joystick positions are each denoted by 4 quarter-quadrants. As a joystick is moved, the corresponding quarter-quadrant becomes filled with color. The fill is proportional to the numeric value associated with the joystick position. Up maximum and Left maximum is -128 while Down maximum and Right maximum is +128. Up and Left are Red while Down and Right are Green. For intermediate directions, both corresponding quarter-quadrants show respective colors.

SPECIAL NOTE: The sole exception to the binary states of the buttons is the L1 button. L1 can initiate 4 states; 0,1,2,3. L1 MUST BE PRESSED TWICE—GREEN TO GRAY AND THEN GRAY TO GREEN—TO ADVANCE THE STATE. Once the state reaches State 3, further presses cause the state to descend back to State 0 and so on. The State is shown on the Pmode display on the lower portion of the GUI, to the right of the display of the list of data received from the Python Client.

The 4 States initiated by L1, which can easily be increased further in the code, dramatically increase the possible number of system modes available to the user.

ESP32 CAMERA SERVER

The reader is referred to the Browser Streaming System and the Python Streaming System diagrams previously shown in Figure 1 and Figure 2 respectively.

What follows is a general description of the Browser System Server and the Python Streaming System Server, outlining their operation and summarizing similarities and differences from a tops-down viewpoint. The step-by-step description and code for each program are given in Part 2, the program section of this paper.

The programs for each of these servers employ the following header files: `ESPAsyncWebServer.h` and `AsyncTCP.h`.

BROWSER STREAMING SERVER and PYTHON STREAMING SERVER: GENERAL DESCRIPTIONS

The Browser *Streaming Server*, *Python_Robot_Browser Server_P*, employs the above header files directly as there is no conflict with the browser header: *WiFiClientSecure.h.*

The Python *Robot_Streaming_Server* to be described next, uses a modified version of the *ESPAsyncWebServer.h* file renamed: *ESPAsyncWebServer_ARS.h* to avoid multiple references with another header *esp_http_server.h* which is used for the Python Streaming Server.

The *ESPAsyncWebServer.h* and the *ESPAsyncWebServer_ARS.h* provides *onWsEvent* handlers which handle messages sent by the Xbox Controller Program and handle the return messages sent back to the Xbox Controller program. Thus, complete communication is established between the Xbox Controller and the Servers as pictured in Figure 1 and Figure 2.

WiFiClientSecure.h provides functions for communicating the browser page and video updates to the Browser. Direct TWO-WAY communication between Server and Browser is realized via AJAX browser technology which will be detailed in PART 2.

The browser page uses the `OpenCV.js` library for a limited library of OpenCV functions for computer vision image processing. The results of this processing are sent back to the server via AJAX and ultimately used by the Robot in the Autonomous mode.

PYTHON STREAMING SERVER: GENERAL DESCRIPTION CONTINUED

As mentioned above, the Python Streaming Server, *Robot_Streaming_Server_P.ino*, uses a modified version of the *ESPAsyncWebServer.h* file renamed `ESPAsyncWebServer_ARS.h` to avoid multiple references with another header `esp_http_server.h` which is used for the Python Streaming Server.

The `esp_http_server.h` header provides functions for streaming video to the python client in the Streaming System. It should be noted here that this data flow is ONE-WAY. Data flow from the python client to the server is handled INDIRECTLY via the Xbox Controller; as described in detail in Part 2. A general description of the two simplex communication channels between the python client program and the Xbox Controller program is provided in the Simplex Communication section, with a more detailed description in Part 2, the program section.

XBOX CONTROLLER

INTRODUCTION

This is a general description of the XBOX CONTROLLER as relates to Robotic Systems described in this paper. The specific code associated with each Joystick, Trigger, Button, and Hat of the Controller will be presented in PART 2.

The Streaming Systems are identical except for one important difference. The Browser System does not require that the Browser program communicates with the Xbox Controller program. The Python Streaming System does require such communication.

In this section, the features of the Xbox Controller program for the Browser System will be described first. Following that, the features of the program for the Python System will be covered. The two descriptions will be largely identical except for the above-mentioned communication features. This repetition is done for the convenience of readers who prefer to focus on one or the other system at any one time.

BROWSER STREAMING SYSTEM

In the Xbox Controller program for the Browser System, *Robot_Streaming_XBOX_P.py*, the state of each button, trigger, hat, and joystick is placed in a list, `list0`, described in PART 2, in each cycle of the program and transmitted to the Browser Server as shown in the Browser System Diagram. In each cycle, the Browser Server also sends a data-filled reply to the Xbox Controller program as shown in that diagram. This data is typically Robot Sensor information and is displayed by the GUI.

One of the buttons, L! (shown in PART 2), can be designated as the Autonomous Button. When this button is pressed Green, the Robot can be programmed to ignore received signals, other than L!, from the Xbox Controller and respond only to the AJAX data it is receiving from the Browser—as per the above Browser Server description and the detailed Browser System code description and the code itself in Part 2.

PYTHON STREAMING SYSTEM

In the Xbox Controller program for the Python Streaming System, *Robot_Streaming_XBOX.py* ,the state of each button, trigger, hat, and joystick is placed in a list, `list0`, as shown in Fig. 4 in each cycle of the program and as happens in the Browser System.

However, in the Autonomous mode as determined by button L!, not all `list0` is transmitted to the Streaming System Server (as it is for the Browser System).

This Controller program substitutes 8 elements of data into 8 of the elements of `list0` before it is transmitted to the Streaming System server. The new elements are taken from the simplex communication channel data sent from the python streaming program as depicted in Figure 1 and are the results of OpenCV routines in that program. (The description and details of the simplex communication and the portion of the Xbox Controller program associated with the simplex channel are given below and in PART 2). List0 with these new elements are transmitted to the server and ultimately to the Robot for use in the Autonomous mode.

As an additional and novel feature of this Python Streaming System, a second simplex channel is used to transmit Controller elements, such as button presses, to the python streaming program to control OpenCV functions during Autonomous mode.

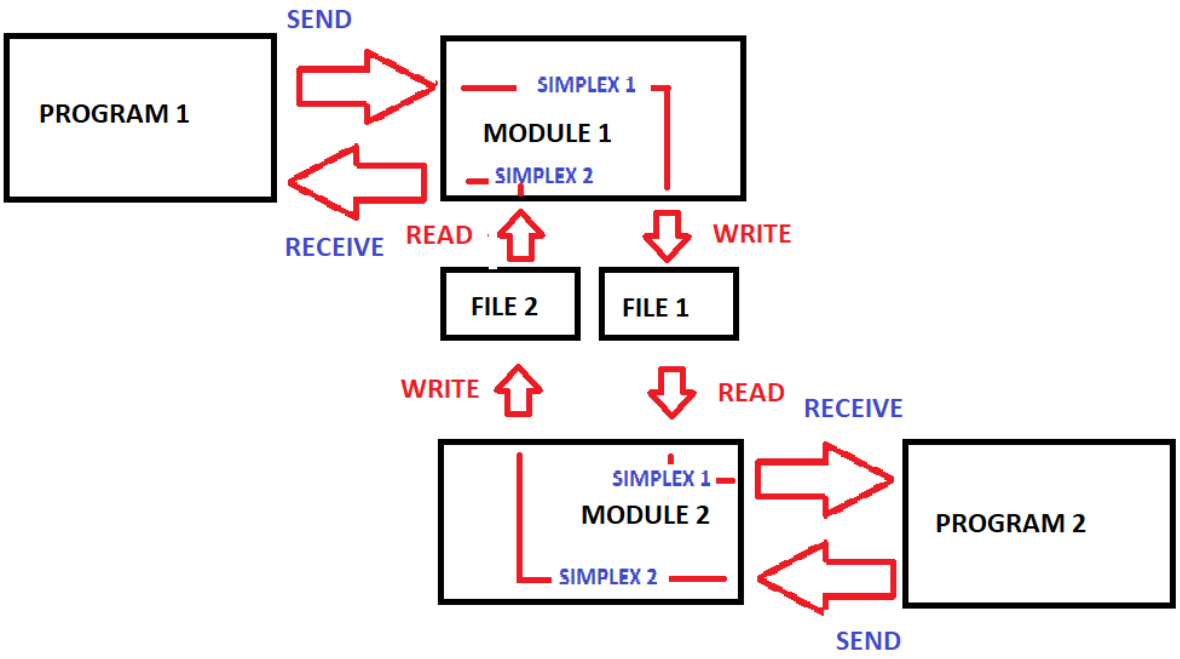
TRADITIONALLY, TRACKBARS ARE USED TO CONTROL OPENCV FUNCTIONS; HERE WE USE AN XBOX CONTROLLER.

Such control is described in detail in upcoming sections of this work.

SIMPLEX COMMUNICATION

Figure 2 showed the two simplex communication channels between the Xbox Controller program and the python streaming window program. These channels were referenced in the above sections. What follows is a general description of these channels to prepare the reader for the detailed description and the code in PART 2.

The simplex channels are conceptually and practically quite simple. Two files are used for temporary data storage. Each channel consists of a write function for one of the two files and a read function for the other of the two files. For example, channel #1 is used to write into file #1 and read from file #2 while channel #2 is used to write into file #2 and read from file #1.



Program 1 (the Xbox Controller, for example) uses Simplex Channel 1 and writes data into File 1 and Program 2 (the python streaming program, for example) reads that data from File 1 in that same channel.

Concurrently, Program 2 uses Simplex channel 2 to write data into File 2 and that data is read by Program 1.

The two simplex channels are packaged into modules and each module is imported into one of the two programs as shown in the figure above. Program 1 imports Module 1 and Program 2 imports Module 2.

In each program cycle, Program 1 writes into File 1 and reads from File 2. Similarly, in each program cycle, Program 2 writes into File 2 and reads from File 1.

The two programs run concurrently so there is a small but finite probability that a channel conflict can occur. However, the conflict probability is small enough that the use of “try”-“except” in each module renders the conflict invisible to the user.

PART 2

SPECIFIC PROGRAM DESCRIPTIONS & THE PROGRAMS

DOWNLOAD PROGRAMS

All programs described and listed below can also be found on GitHub:

- BROWSER STREAMING SYSTEM:
  - https://github.com/arshacker/ESP32-ROBOT-CAM-BROWSER-XBOX-CONTROLLER-OPENCV-SYSTEM
- PYTHON STREAMING SYSTEM:
  - https://github.com/arshacker/ESP32-ROBOT-CAM-VIDEO-XBOX-CONTROLLER-OPENCV-SYSTEM

SOFTWARE PROGRAM DEVELOPMENT ENVIRONMENTS

Three development environments are needed for a reader interested in both the Browser and the Streaming Systems described in this paper.

- Arduino IDE: Browser and Streaming System
- Python IDLE: Browser and Streaming System
- Chrome Web Browser and an HTML Editor: Browser System

ARDUINO IDE

As far as the ESP32 is concerned in this project, the Arduino IDE is used exclusively. Version 1.8.2 was used to develop the programs in this paper. Installing ESP32-specific libraries into the Arduino IDE is covered

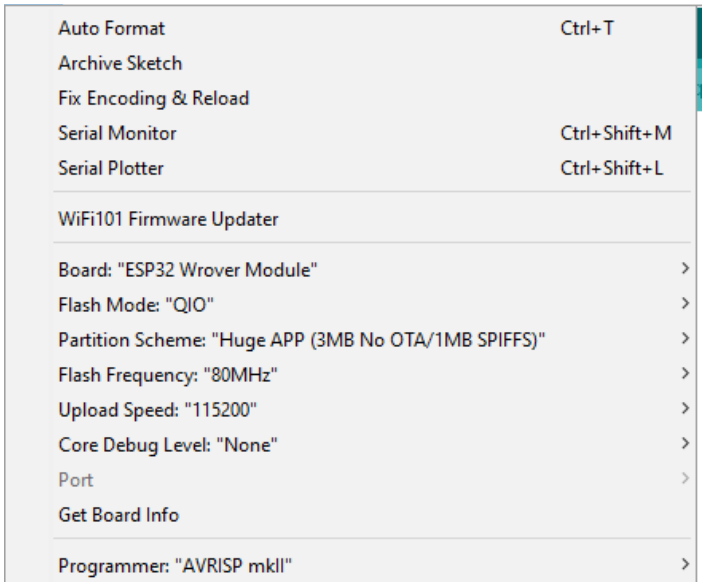
in the Random Nerds Tutorials in detail for those readers who have no experience with the ESP32.

- https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/

Aficionados of VS Code, who prefer it to the Arduino IDE, please refer to:

- https://randomnerdtutorials.com/vs-code-platformio-ide-esp32-esp8266-arduino/

Once all installations are completed and the specific program is opened, the following settings should be used. The port selected will be specific to the user.



PYTHON IDLE

Python 3.6.5 has been used to develop the Streaming Window program for the Streaming System shown in Fig. 1 and the XBOX Controller programs for the Streaming System in Fig. 1 and the Browser System shown in Fig. 2.

Important note #1: in the streaming system, the two python programs must be run as separate instances. This is done by simply opening 2 instances of idle and then using one of the two instances of idle for each program.

Important note #2: in both systems, the Xbox controller GUI window must be an active window (selected with the mouse) so that the controller inputs are registered by the program.

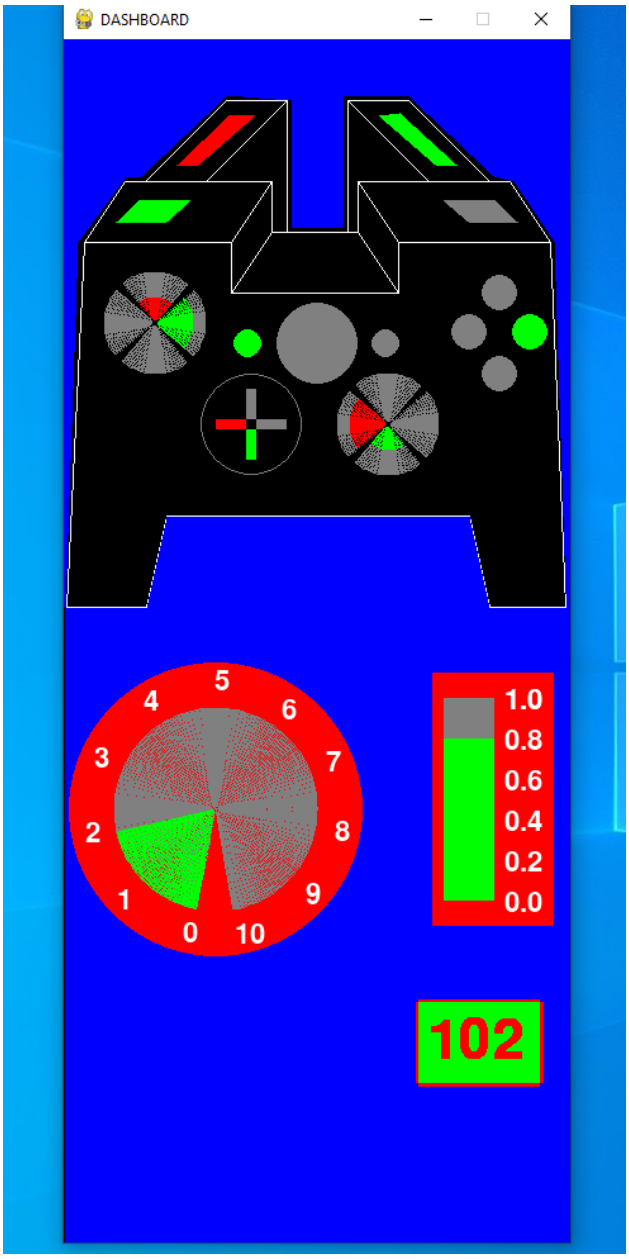
CHROME BROWSER

The author has used the CHROME web browser exclusively during the development of the Browser System. Debugging during development used **CTRL**-**SHIFT**-**J** to see embedded messages in the Browser Window.

XBOX CONTROLLER GUI

dashboard_26_module.py

The Dashboard Module is the GUI for the Xbox Controller. A screenshot of the GUI is shown below. It is a render of the XBOX Controller with Buttons, Triggers, Hat, and two Joysticks.



The two Joysticks are each represented by 4 quadrant arcs positioned as Upper, Lower, Left, and Right. At every drawing of the GUI, the quadrants are colored as follows: upper and left (negative values of joysticks) are solid RED and lower and right (positive values of joysticks) are solid GREEN. The position of a Joystick from the center immediately colors the corresponding quadrant arc GRAY starting at the top of the quadrant. The amount of GRAY colored leaves the remainder of RED or GREEN to denote the position of the Joystick. This is DEMONSTRATED HERE by the Right Joystick (Joystick1) where that Joystick position is partially Left and partially Down. Similarly, Joystick2 is partially Right and partially Up.

The initial view of the Controller is GRAY for all elements except for the Triggers, which are RED. The latter departure has historically for this project, been to serve as a warning (if RED color does NOT occur) that a system runtime error has occurred. The GRAY for elements other than Triggers represents a logic 0 while the GREEN represents a logic 1. For the Triggers the RED is logic 0 and the GREEN is logic 1.

(In the Xbox Controller program described previously the Button, and Trigger inputs are latched to facilitate operation by the Controller user; namely, press to change state and another required press to change state again.)

Sensor widgets are drawn below the Controller picture with examples of sensor readings.

- **#ANN:1** is the listing of dimensions; the vast majority of which are expressed as a fraction of the screen width (250).
- **#ANN:2** show the Dashboard function. This function draws
    - Black Fill of entire screen
    - White Line Drawing of XBox Controller
    - Buttons, Triggers, Hat, Joysticks
    - Blue Background*
    - Sensors*
- #**ANN:3** shows the functions that draw the elements of the Controller. (Some of the functions called by the program were only used for development purposes and are not used by the Controller that imports and uses the module.)

* Sensors drawing must follow after Blue Background. The order of other elements is not critical.

DASHBOARD_26_MODULE.py CODE

- https://github.com/arshacker/ESP32-ROBOT-CAM-BROWSER-XBOX-CONTROLLER-OPENCV-SYSTEM/blob/main/dashboard_26_module.py

```
import pygame
from pygame.locals import *

import math

pygame.init()

ScreenWidth = 400
ScreenHeight = 950

ScreenSize = (ScreenWidth,ScreenHeight)
```

```
DashScreen = pygame.display.set_mode(ScreenSize)
pygame.display.set_caption("DASHBOARD")

running = True

black = (0,0,0)
gray = (128,128,128)
light_gray = (200,200,200)
dark_gray = (64,64,64)
white = (255,255,255)
red = (255,0,0)
light_red =(128,0,0)
green = (0,255,0)
blue = (0,0,255)
#ANN:1
LineWidth = 1
LWC=1   #LWC is Line Width correction of blue background next to black outline of
xbox
Offset = ScreenWidth*(10/250)                      #10
XOrigin = 0+LWC
YOrigin = ScreenWidth*(280/250)                      #ScreenWidth*1.12
#YOrigin = ScreenWidth*1.12
Slant = ScreenWidth*(100/250)          #100
Handle = ScreenWidth*(40/250)         #40
HWratio=ScreenWidth*(1.12/250)      #1.12
HWIratio=ScreenWidth*(0.95/250)     #0.95
Kappa = ScreenWidth*(45/250)
DepthX=ScreenWidth*(20/250)         #20
DepthY=ScreenWidth*(30/250)        #30
TrigX=ScreenWidth*(50/250)          #50
TrigY=ScreenWidth*(40/250)          #40
TrigW=ScreenWidth*(30/250)         #30
TrigXO=ScreenWidth*(30/250)         #30
Del=ScreenWidth*(8/250)           #8
Alpha=ScreenWidth*(80/250)        #89
Radius_ABXY= ScreenWidth*(9/250)  #9
Beta = ScreenWidth*(25/250)    #25
#-----------JOYSTICKS-----------------
xCenter_1 = ScreenWidth*(160/250)    #160
yCenter_1 = ScreenWidth*(140/250)    #140
radius_1 = math.floor((ScreenWidth/250)*25)    #25
xCenter_2 = ScreenWidth*(45/250)    #160
yCenter_2 = ScreenWidth*(90/250)    #140
radius_2 = math.floor((ScreenWidth/250)*25)    #25
radius_3 = 5
radius_4 = 10
radius_5 = 25
#--------end JOYSTICKS------------
x_HatOrigin = ScreenWidth*(90/250)     #90
y_HatOrigin= ScreenWidth*(193/250) #195
h_width= ScreenWidth*(5/250)  #5
h_height= ScreenWidth*(15/250) #15
L1_R1_radius = ScreenWidth*(7/250)  #7
x_ZOrigin = ScreenWidth*(125/250)   #125
```

```
y_ZOrigin = ScreenWidth*(150/250)   #150
Z_radius = ScreenWidth*(1.8/250)*L1_R1_radius    #1.8
#------------SENSORS-------------------------
ZEROSCALEANGLE =260
TOTALANGLE = 335
FontSize = math.floor(ScreenWidth*(40/250))  #40
FontSize_1 = math.floor(ScreenWidth*(20/250))  #20
FontSize_2 = math.floor(ScreenWidth*(20/250))  #20

BottomSensor_2 = ScreenWidth*1.5-2*radius_1 + 2*2*radius_1
Y_Sensor_3 = BottomSensor_2 + 2*radius_1
DelX_Sensor_3 = math.floor((ScreenWidth/250)*60)  #60
DelY_Sensor_3 = math.floor((ScreenWidth/250)*40)  #40
YCenter_Sensor_3 = math.floor((ScreenWidth/250)*6)          #6
XCenter_Sensor_3 = math.floor((ScreenWidth/250)*5)          #5
YCORR_Sensor_1 = math.floor((ScreenWidth/250)*5)          #5
NUMCORR_Sensor_1 = math.floor((ScreenWidth/250)*15)          #15
NUMCORR2_Sensor_1 = math.floor((ScreenWidth/250)*10)          #10
NUMCORR3_Sensor_1 = math.floor((ScreenWidth/250)*12)          #12
NUMCORR4_Sensor_1 = math.floor((ScreenWidth/250)*6)          #6
NUMCORR5_Sensor_1 = math.floor((ScreenWidth/250)*3)          #3
NUMBACKGRND_Sensor_2 = math.floor((ScreenWidth/250)*6)          #6
NUMCORR_Sensor_2 = math.floor((ScreenWidth/250)*6)          #6

DefaultFont = None
DashFont = pygame.font.Font(DefaultFont,FontSize)
DashFont_1 = pygame.font.Font(DefaultFont,FontSize_1)
DashFont_2 = pygame.font.Font(DefaultFont,FontSize_2)
#------------END SENSORS------------------
running = True
#ANN:2
def Dashboard():   #draw black background. draw xbox with white lines. draw lower
baclground
                   #and upper background using blue polygons

    DashScreen.fill(black)
    XBOX_Outline()
    Button_B(gray)
    Button_X(gray)
    Button_Y(gray)
    Button_A(gray)
    background1(blue)
    background2(blue)
    TrigL_Sig(gray)
    TrigR_Sig(gray)
    Button_L(gray)
    Button_R(gray)
    #print(Angle_to_Rads(180))
    #anArc(gray,160,140,25,0,180)
    JStick_1(red,green,red,green)
    JStick_2(red,green,red,green)
    Hat(gray,gray,gray,gray)
    #pygame.draw.arc(DashScreen,gray,(200,250,50,50),0,3.14,25)
    ZbuttonGroup(gray,gray,gray)
```

```
############WIDGETS#############
    SENSOR_1_NUM_BACKGRND(red)
    SENSOR_1(red)
    SENSOR_1_ARC0(red,260,-80)
    SENSOR_1_FILL(gray,-80,260)
    ####SENSOR_1_VALUE(green,-80,260)
    #FULL SCALE   STARTANGLE=-80   END ANGLE=260
    #THUS   FULLSCALEANGLE=-80     ZEROSCALEANGLE= 260
    #TOTALANGLE=340
    #ZERO SCALE   STARTANGLE=260   ENDANGLE=260
    ####SENSOR_1_VALUE(green,260-0.7*340,260)
    #SENSOR_1_VALUE(green,0.7)
    SENSOR_1_NUM(white)
    SENSOR_2_NUM_BACKGROUND(red)
    SENSOR_2(red)
    SENSOR_2_FILL(gray)
    #SENSOR_2_VALUE(green,0.7)
    SENSOR_2_NUM(white)
    SENSOR_3(red)
    SENSOR_3_FILL(green)
    ###SENSOR_3_VALUE(red,127,ScreenWidth*.7+XCenter_Sensor_3,Y_Sensor_3 +
YCenter_Sensor_3)
    #SENSOR_3_VALUE(red,127)
    pygame.display.update()

##########################DRAWING NOTES##########################

#ARC START ANGLE=0 DEGREES   X=X0,Y=0 END ANGLE IS CCW
#TRIG FUNCS degrees(value),radians(degrees),pi,sin and cos in radians

########################END DRAWING NOTES########################

###################SENSOR WIDGETS#######################################
#ANN:3
def SENSOR_1_NUM_BACKGRND(color):
    pygame.draw.circle(DashScreen,color,(ScreenWidth*0.3,ScreenWidth*1.5+YCORR_Se
nsor_1 ),2.8*radius_1+4*LWC)

def SENSOR_1(color):
    pygame.draw.circle(DashScreen,color,(ScreenWidth*0.3,ScreenWidth*1.5+YCORR_Se
nsor_1 ),2*radius_1+4*LWC,4)

def SENSOR_1_VALUE(color,decFraction):
    pygame.draw.arc(DashScreen,color,(ScreenWidth*0.30-
2*radius_1,ScreenWidth*1.5-2*radius_1+YCORR_Sensor_1 ,\
                    2*2*radius_1,2*2*radius_1),\
                    Angle_to_Rads(ZEROSCALEANGLE-
decFraction*TOTALANGLE),Angle_to_Rads(ZEROSCALEANGLE),2*radius_1)

#def SENSOR_1_VALUE(color,startAngle,endAngle):
#    pygame.draw.arc(DashScreen,color,(ScreenWidth*0.25-
2*radius_1,ScreenWidth*1.5-2*radius_1,\
#   2*2*radius_1,2*2*radius_1),\
#Angle_to_Rads(startAngle),Angle_to_Rads(endAngle),2*radius_1)
```

```
def SENSOR_1_ARC0(color,startAngle,endAngle):
    pygame.draw.arc(DashScreen,color,(ScreenWidth*0.30-
2*radius_1,ScreenWidth*1.5-2*radius_1+YCORR_Sensor_1 ,\
                    2*2*radius_1,2*2*radius_1),\
                    Angle_to_Rads(startAngle),Angle_to_Rads(endAngle),2*radius_1)


def SENSOR_1_FILL(color,startAngle,endAngle):
    pygame.draw.arc(DashScreen,color,(ScreenWidth*0.3-2*radius_1,ScreenWidth*1.5-
2*radius_1+YCORR_Sensor_1 ,\
                    2*2*radius_1,2*2*radius_1),\
                    Angle_to_Rads(startAngle),Angle_to_Rads(endAngle),2*radius_1)


def SENSOR_1_NUM(color):
    astring = str(0)
    b_numb = astring.encode()
    DashTextGraphic_1 = DashFont_1.render(b_numb,True,color)
    DashScreen.blit(DashTextGraphic_1,(ScreenWidth*0.3-
(2*radius_1+NUMCORR3_Sensor_1)*math.sin(math.pi/12),\
                                       ScreenWidth*1.5+(2*radius_1+NUMCORR3_Sensor_
1)*math.cos(math.pi/12)))
    for numb in [1,2,3,4,5]:
        astring = str(numb)
        b_numb = astring.encode()
        DashTextGraphic_1 = DashFont_1.render(b_numb,True,color)
        DashScreen.blit(DashTextGraphic_1,\
               (ScreenWidth*0.3-
(2*radius_1+NUMCORR_Sensor_1)*math.sin(math.pi/12+((math.pi-
math.pi/12)/5)*numb),\
                ScreenWidth*1.5+(2*radius_1+NUMCORR_Sensor_1)*math.cos(math.pi/12
+((math.pi-math.pi/12)/5)*numb)))

    for numb in [1,2,3,4,5]:
        astring = str(numb+5)
        b_numb = astring.encode()
        DashTextGraphic_1 = DashFont_1.render(b_numb,True,color)
        if numb!=5:
            DashScreen.blit(DashTextGraphic_1,\
              (ScreenWidth*0.3-
(2*radius_1+NUMCORR2_Sensor_1)*math.sin(math.pi+((math.pi-math.pi/12)/5)*numb),\
               ScreenWidth*1.5+(2*radius_1+NUMCORR2_Sensor_1)*math.cos(math.pi+((
math.pi-math.pi/12)/5)*numb)))

        if numb==5:
            DashScreen.blit(DashTextGraphic_1,\
               (ScreenWidth*0.3-
(2*radius_1+NUMCORR2_Sensor_1)*math.sin(math.pi+((math.pi-math.pi/12)/5)*numb)-
NUMCORR4_Sensor_1,\
                ScreenWidth*1.5+(2*radius_1+NUMCORR2_Sensor_1)*math.cos(math.pi+(
(math.pi-math.pi/12)/5)*numb)+NUMCORR5_Sensor_1))

def SENSOR_2_NUM_BACKGROUND(color):
```

```
    pygame.draw.rect(DashScreen,color,(ScreenWidth*.75-
NUMBACKGRND_Sensor_2,ScreenWidth*1.5-2.5*radius_1,\
                                       2.4*radius_1,2.5*2*radius_1))

def SENSOR_2(color):
    pygame.draw.rect(DashScreen,color,(ScreenWidth*.75,ScreenWidth*1.5-
2*radius_1,radius_1,2*2*radius_1),6)

def SENSOR_2_FILL(color):
    pygame.draw.rect(DashScreen,color,(ScreenWidth*.75,ScreenWidth*1.5-
2*radius_1,radius_1,2*2*radius_1))

def SENSOR_2_VALUE(color,decFraction):
    pygame.draw.rect(DashScreen,color,(ScreenWidth*.75,\
                         ScreenWidth*1.5-2*radius_1 + (1-
decFraction)*(2*2*radius_1),\
                         radius_1,\
                         (decFraction)*2*2*radius_1))

def SENSOR_2_NUM(color):
    astring = str(1.0)
    b_numb = astring.encode()
    DashTextGraphic_1 = DashFont_1.render(b_numb,True,color)
    DashScreen.blit(DashTextGraphic_1,(ScreenWidth*.75+1.2*radius_1,\
                                       ScreenWidth*1.5-2*radius_1-
NUMCORR_Sensor_2))
    for i in [0.0,0.2,0.4,0.6,0.8,1.0]:
        astring = str(i)
        b_numb = astring.encode()
        DashTextGraphic_1 = DashFont_1.render(b_numb,True,color)
        DashScreen.blit(DashTextGraphic_1,(ScreenWidth*.75+1.2*radius_1,\
                                       ScreenWidth*1.5-2*radius_1-
NUMCORR_Sensor_2+(2*2*radius_1-i*2*2*radius_1)))


#DashScreen.blit(DashTextGraphic_1,(ScreenWidth*.75+1.2*radius_1,\
#ScreenWidth*1.5-2*radius_1,radius_1,2*2*radius_1))


def SENSOR_3(color):
    pygame.draw.rect(DashScreen,color,(ScreenWidth*.7,Y_Sensor_3,DelX_Sensor_3,De
lY_Sensor_3),6)

def SENSOR_3_FILL(color):
    pygame.draw.rect(DashScreen,color,(ScreenWidth*.7,Y_Sensor_3,DelX_Sensor_3,De
lY_Sensor_3))

def SENSOR_N_FILL(color):
    pygame.draw.rect(DashScreen,color,(0,840,400,70))

def SENSOR_3_VALUE(color,numb):
    #DefaultFont = None
    #GameFont = pygame.font.Font(DefaultFont,60)
    astring = str(numb)
```

```
    b_numb = astring.encode()
    DashTextGraphic = DashFont.render(b_numb,True,color)
    DashScreen.blit(DashTextGraphic,(ScreenWidth*.7+XCenter_Sensor_3,Y_Sensor_3 +
YCenter_Sensor_3))

def SENSOR_N_VALUE(color,numb,xloc,yloc):
    bstring = str(numb)
    c_numb = bstring.encode()
    DashTextGraphic2 = DashFont_2.render(c_numb,True,color)
    DashScreen.blit(DashTextGraphic2,(xloc,yloc))


##################END SENSOR WIDGETS#################################

def XBOX_Outline():    #lines start lower left and proceed clockwise in all
routines in this function
    Outline_Front = pygame.draw.lines(DashScreen,white,True,\
                                #1              #2
                        [(XOrigin+1*LWC,YOrigin),(Offset,Slant),\
                         #(ScreenWidth-Offset,Slant),\
                          (ScreenWidth*0.33,Slant),(ScreenWidth*0.33,Slant+Beta),
\
                          (ScreenWidth*0.66,Slant+Beta),(ScreenWidth*0.66,Slant),
\
                          (ScreenWidth-Offset,Slant),\
                                      #14
                         (ScreenWidth-4*LWC,YOrigin),\
                         (ScreenWidth-Handle,YOrigin),\
                         (ScreenWidth-Handle-Offset,YOrigin-Kappa),\
                         (XOrigin+Handle+Offset,YOrigin-Kappa),\
                         (XOrigin+Handle,YOrigin)],LineWidth)
                         #(XOrigin+Handle,ScreenWidth*HWratio)],LineWidth)

                                  #15  (ScreenWidth,0)
                                  #16  (0,0)

    #Used for initial design of channel in drawing
    #Channel_Front = pygame.draw.lines(DashScreen,white,False,\
    #                     [(ScreenWidth*0.33,Slant),\
    #                      (ScreenWidth*0.33,Slant+Beta),\
    #                      (ScreenWidth*0.66,Slant+Beta),\
    #                      (ScreenWidth*0.66,Slant)],LineWidth)


    Outline_Depth = pygame.draw.lines(DashScreen,white,False,\
                                                #3
                        [(Offset,Slant),(Offset+DepthX,Slant-DepthY),\

                         (ScreenWidth*0.33+DepthX,Slant-
DepthY),(ScreenWidth*0.33+DepthX,Slant+Beta-DepthY),\
                         (ScreenWidth*0.66-DepthX,Slant+Beta-
DepthY),(ScreenWidth*0.66-DepthX,Slant-DepthY),\
```

```
                                                #12
                          (ScreenWidth-Offset-DepthX,Slant-DepthY),\
                                               #13
                          (ScreenWidth-Offset,Slant)],LineWidth)

    #Used for initial design of channel in drawing
    #Channel_Depth = pygame.draw.lines(DashScreen,white,False,\
    #                     [(ScreenWidth*0.33+DepthX,Slant-
DepthY),(ScreenWidth*0.33+DepthX,Slant+Beta-DepthY),\
    #                      (ScreenWidth*0.66-DepthX,Slant+Beta-
DepthY),(ScreenWidth*0.66-DepthX,Slant-DepthY)],LineWidth)

    Channel_ConnectorL_Top = pygame.draw.lines(DashScreen,white,False,\
                          [(ScreenWidth*0.33,Slant),(ScreenWidth*0.33+DepthX,Slant
-DepthY)],LineWidth)

    Channel_ConnectorL_Bot = pygame.draw.lines(DashScreen,white,False,\
                          [(ScreenWidth*0.33,Slant+Beta),(ScreenWidth*0.33+DepthX,
Slant+Beta-DepthY)],LineWidth)

    Channel_ConnectorR_Bot = pygame.draw.lines(DashScreen,white,False,\
                          [(ScreenWidth*0.66,Slant+Beta),(ScreenWidth*0.66-
DepthX,Slant+Beta-DepthY)],LineWidth)

    Channel_ConnectorR_Top = pygame.draw.lines(DashScreen,white,False,\
                          [(ScreenWidth*0.66,Slant),(ScreenWidth*0.66-
DepthX,Slant-DepthY)],LineWidth)


    Outline_TriggerL = pygame.draw.lines(DashScreen,white,False,\
                          #4                  #5
         [(Offset+TrigXO,Slant-DepthY),(Offset+DepthX+TrigX,Slant-DepthY-TrigY),\
                                    #6
          (Offset+DepthX+TrigX+TrigW,Slant-DepthY-TrigY),\
          (Offset+TrigXO+TrigW,Slant-DepthY)],LineWidth)

    Outline_TriggerL1 = pygame.draw.lines(DashScreen,white,False,\
                                                                  #7
       [(Offset+DepthX+TrigX+TrigW,Slant-DepthY-
TrigY),(Offset+DepthX+TrigX+TrigW,Slant-DepthY+Beta)],LineWidth)

    #TriggerR starts lower right, goes ccw
    Outline_TriggerR = pygame.draw.lines(DashScreen,white,False,\
                        #11                                        #10
    [(ScreenWidth-Offset-TrigXO,Slant-DepthY),(ScreenWidth-Offset-DepthX-
TrigX,Slant-DepthY-TrigY),\
                                    #9
     (ScreenWidth-Offset-DepthX-TrigX-TrigW,Slant-DepthY-TrigY),\
     (ScreenWidth-Offset-TrigXO-TrigW,Slant-DepthY)],LineWidth)

    Outline_TriggerR1 = pygame.draw.lines(DashScreen,white,False,\
        [(ScreenWidth-Offset-DepthX-TrigX-TrigW,Slant-DepthY-TrigY),\
                                    #8
        (ScreenWidth-Offset-DepthX-TrigX-TrigW,Slant-DepthY+Beta)],LineWidth)
```

```
def Button_L(color):
    pygame.draw.polygon(DashScreen,color,\
    [(Offset+2*Del,Slant-1.2*Del),(Offset+DepthX+0.9*Del,Slant-DepthY+1.2*Del),\
     (Offset+DepthX+0.4*Alpha,Slant-DepthY+1.2*Del),(Offset+0.5*Alpha,Slant-
1.2*Del)],0)

def Button_R(color):
    pygame.draw.polygon(DashScreen,color,\
    [(ScreenWidth-Offset-2*Del,Slant-1.2*Del),(ScreenWidth-Offset-DepthX-
0.9*Del,Slant-DepthY+1.2*Del),\
     (ScreenWidth-Offset-DepthX-0.4*Alpha,Slant-DepthY+1.2*Del),(ScreenWidth-
Offset-0.5*Alpha,Slant-1.2*Del)],0)


def TrigL_Sig(color):
    pygame.draw.polygon(DashScreen,color,\
         [(Offset+TrigXO+2.0*Del,Slant-DepthY-
Del),(Offset+DepthX+TrigX+0.2*Del,Slant-DepthY-TrigY+Del),\
          (Offset+DepthX+TrigX+TrigW-2*Del,Slant-DepthY-TrigY+Del),\
          (Offset+TrigXO+TrigW-0.5*Del,Slant-DepthY-Del)],0)

def TrigR_Sig(color):
    pygame.draw.polygon(DashScreen,color,\
         [(ScreenWidth-Offset-TrigXO-2.0*Del,Slant-DepthY-1.0*Del),(ScreenWidth-
Offset-DepthX-TrigX-0.5*Del,Slant-DepthY-TrigY+0.8*Del),\
          (ScreenWidth-Offset-DepthX-TrigX-TrigW+2.0*Del,Slant-DepthY-
TrigY+1.0*Del),\
          (ScreenWidth-Offset-TrigXO-TrigW+0.5*Del,Slant-DepthY-
1.0*Del)],0)

def background1(color):  #area below xbox. polyg starts lower left corner of xbox
and proceeds clockwise
    pygame.draw.polygon(DashScreen,color,[(XOrigin,ScreenHeight),(XOrigin,YOrigin
+LWC),\
                (XOrigin+Handle,YOrigin+LWC),\
                (XOrigin+Handle+Offset,YOrigin-Kappa+LWC),\
                (ScreenWidth-Handle-Offset,YOrigin-Kappa+LWC),\
                (ScreenWidth-Handle,YOrigin+LWC),\
                (ScreenWidth,YOrigin+LWC),\
                (ScreenWidth,ScreenHeight)],0)

def background2(color):   #area above xbox. polyg starts lower left corner of
xbox and proceeds counter clockwise. nbrs below correspond to nbrs above.
                                                  #1                #2
    pygame.draw.polygon(DashScreen,color,[(XOrigin-LWC,YOrigin),(Offset-
6*LWC,Slant),\
                          #3
         (Offset+DepthX,Slant-DepthY-4*LWC),\
                            #4                                  #5
         (Offset+TrigXO-4*LWC,Slant-DepthY-4*LWC),(Offset+DepthX+TrigX,Slant-
DepthY-TrigY-4*LWC),\
                                       #6
         (Offset+DepthX+TrigX+TrigW+4*LWC,Slant-DepthY-TrigY),\
```

```
                                #7                                      #8

        (Offset+DepthX+TrigX+TrigW+4*LWC,Slant-DepthY-4*LWC+Beta),(ScreenWidth-
Offset-DepthX-TrigX-TrigW-4*LWC,Slant-DepthY-4*LWC+Beta),\
                                        #9
        (ScreenWidth-Offset-DepthX-TrigX-TrigW-4*LWC,Slant-DepthY-TrigY-4*LWC),\
                                        #10
        (ScreenWidth-Offset-DepthX-TrigX,Slant-DepthY-TrigY-4*LWC),\
                                    #11                                 #12
        (ScreenWidth-Offset-TrigXO+4*LWC,Slant-DepthY-4*LWC),(ScreenWidth-
Offset-DepthX+4*LWC,Slant-DepthY),\
                               #13                          #14
        (ScreenWidth-
Offset+4*LWC,Slant),(ScreenWidth+6*LWC,ScreenWidth*HWratio),\
              #15        #16
        (ScreenWidth,0),(0,0)],0)

def Button_B(color):
    pygame.draw.circle(DashScreen,color,(ScreenWidth*0.92,ScreenWidth*0.58),Radiu
s_ABXY)

def Button_X(color):
    pygame.draw.circle(DashScreen,color,(ScreenWidth*0.8,ScreenWidth*0.58),Radius
_ABXY)

def Button_Y(color):
    pygame.draw.circle(DashScreen,color,(ScreenWidth*0.86,ScreenWidth*0.5),Radius
_ABXY)

def Button_A(color):
    pygame.draw.circle(DashScreen,color,(ScreenWidth*0.86,ScreenWidth*0.66),Radiu
s_ABXY)

#0 degrees is straight up and degrees go ccw
def Angle_to_Rads(angle):
    return round(angle*(6.28/360),2)

def anArc(color,xCenter,yCenter,radius,startAngle,endAngle):
    pygame.draw.arc(DashScreen,color,(xCenter-
radius,yCenter+radius,2*radius,2*radius),\
                    Angle_to_Rads(startAngle),Angle_to_Rads(endAngle),radius)

def JStick_1(color_up,color_down,color_left,color_right):
    pygame.draw.arc(DashScreen,color_up,(xCenter_1-
radius_1,yCenter_1+radius_1,2*radius_1,2*radius_1),\
                    Angle_to_Rads(50),Angle_to_Rads(130),radius_1)

    pygame.draw.arc(DashScreen,color_left,(xCenter_1-
radius_1,yCenter_1+radius_1,2*radius_1,2*radius_1),\
                    Angle_to_Rads(140),Angle_to_Rads(220),radius_1)

    pygame.draw.arc(DashScreen,color_down,(xCenter_1-
radius_1,yCenter_1+radius_1,2*radius_1,2*radius_1),\
```

```
                    Angle_to_Rads(230),Angle_to_Rads(310),radius_1)

    pygame.draw.arc(DashScreen,color_right,(xCenter_1-
radius_1,yCenter_1+radius_1,2*radius_1,2*radius_1),\
                    Angle_to_Rads(320),Angle_to_Rads(40),radius_1)

def JStick_1Y(arc_width_up,arc_width_down):
    pygame.draw.arc(DashScreen,gray,(xCenter_1-
radius_1,yCenter_1+radius_1,2*radius_1,2*radius_1),\
                    Angle_to_Rads(50),Angle_to_Rads(130),arc_width_up)
    pygame.draw.arc(DashScreen,gray,(xCenter_1-
radius_1,yCenter_1+radius_1,2*radius_1,2*radius_1),\
                    Angle_to_Rads(230),Angle_to_Rads(310),arc_width_down)

def JStick_1X(arc_width_left,arc_width_right):
    pygame.draw.arc(DashScreen,gray,(xCenter_1-
radius_1,yCenter_1+radius_1,2*radius_1,2*radius_1),\
                    Angle_to_Rads(140),Angle_to_Rads(220),arc_width_left)
    pygame.draw.arc(DashScreen,gray,(xCenter_1-
radius_1,yCenter_1+radius_1,2*radius_1,2*radius_1),\
                    Angle_to_Rads(320),Angle_to_Rads(40),arc_width_right)

def JStick_2(color_up,color_down,color_left,color_right):
    pygame.draw.arc(DashScreen,color_up,(xCenter_2-
radius_2,yCenter_2+radius_2,2*radius_2,2*radius_2),\
                    Angle_to_Rads(50),Angle_to_Rads(130),radius_2)

    pygame.draw.arc(DashScreen,color_left,(xCenter_2-
radius_2,yCenter_2+radius_2,2*radius_2,2*radius_2),\
                    Angle_to_Rads(140),Angle_to_Rads(220),radius_2)

    pygame.draw.arc(DashScreen,color_down,(xCenter_2-
radius_2,yCenter_2+radius_2,2*radius_2,2*radius_2),\
                    Angle_to_Rads(230),Angle_to_Rads(310),radius_2)

    pygame.draw.arc(DashScreen,color_right,(xCenter_2-
radius_2,yCenter_2+radius_2,2*radius_2,2*radius_2),\
                    Angle_to_Rads(320),Angle_to_Rads(40),radius_2)

def JStick_2Y(arc_width_up,arc_width_down):
    pygame.draw.arc(DashScreen,gray,(xCenter_2-
radius_2,yCenter_2+radius_2,2*radius_2,2*radius_2),\
                    Angle_to_Rads(50),Angle_to_Rads(130),arc_width_up)
    pygame.draw.arc(DashScreen,gray,(xCenter_2-
radius_2,yCenter_2+radius_2,2*radius_2,2*radius_2),\
                    Angle_to_Rads(230),Angle_to_Rads(310),arc_width_down)

def JStick_2X(arc_width_left,arc_width_right):
    pygame.draw.arc(DashScreen,gray,(xCenter_2-
radius_2,yCenter_2+radius_2,2*radius_2,2*radius_2),\
                    Angle_to_Rads(140),Angle_to_Rads(220),arc_width_left)
    pygame.draw.arc(DashScreen,gray,(xCenter_2-
radius_2,yCenter_2+radius_2,2*radius_2,2*radius_2),\
                    Angle_to_Rads(320),Angle_to_Rads(40),arc_width_right)
```

```
#   x->  y down pos
#
#
#             RECTANGLE--SEE HAT BELOW
#
#                    xo,yo....
#                         .  .
#                         .  .
#               xo,yo+h    ....xo+w,yo+h
#



def Hat(color_up,color_down,color_left,color_right):
    pygame.draw.circle(DashScreen,gray,(x_HatOrigin+(h_width/2),y_HatOrigin-
(h_width/2)),h_height+2*h_width,1)
    pygame.draw.rect(DashScreen,color_down,(x_HatOrigin,y_HatOrigin,h_width,h_hei
ght))
    pygame.draw.rect(DashScreen,color_up,(x_HatOrigin,y_HatOrigin-h_width-
h_height,h_width,h_height))
    pygame.draw.rect(DashScreen,color_right,(x_HatOrigin+h_width,y_HatOrigin-
h_width,h_height,h_width))
    pygame.draw.rect(DashScreen,color_left,(x_HatOrigin-h_height,y_HatOrigin-
h_width,h_height,h_width))

def HatY(color_up,color_down):
    pygame.draw.circle(DashScreen,gray,(x_HatOrigin+(h_width/2),y_HatOrigin-
(h_width/2)),h_height+2*h_width,1)
    pygame.draw.rect(DashScreen,color_down,(x_HatOrigin,y_HatOrigin,h_width,h_hei
ght))
    pygame.draw.rect(DashScreen,color_up,(x_HatOrigin,y_HatOrigin-h_width-
h_height,h_width,h_height))

def HatX(color_left,color_right):
    pygame.draw.circle(DashScreen,gray,(x_HatOrigin+(h_width/2),y_HatOrigin-
(h_width/2)),h_height+2*h_width,1)
    pygame.draw.rect(DashScreen,color_right,(x_HatOrigin+h_width,y_HatOrigin-
h_width,h_height,h_width))
    pygame.draw.rect(DashScreen,color_left,(x_HatOrigin-h_height,y_HatOrigin-
h_width,h_height,h_width))


def ZbuttonGroup(color_left,color_right, color_center):
    pygame.draw.circle(DashScreen,color_left,(x_ZOrigin-Z_radius-
2*L1_R1_radius,y_ZOrigin),L1_R1_radius)
    pygame.draw.circle(DashScreen,color_right,(x_ZOrigin+Z_radius+2*L1_R1_radius,
y_ZOrigin),L1_R1_radius)
    pygame.draw.circle(DashScreen,color_center,(x_ZOrigin,y_ZOrigin),Z_radius)

def Button_L1(color):
    pygame.draw.circle(DashScreen,color,(x_ZOrigin-Z_radius-
2*L1_R1_radius,y_ZOrigin),L1_R1_radius)
```

```
def Button_R1(color):
    pygame.draw.circle(DashScreen,color,(x_ZOrigin+Z_radius+2*L1_R1_radius,y_ZOri
gin),L1_R1_radius)

def main():
    global running
    Dashboard()
    while (running):
        #BUTTONS AND TRIGGERS EXAMPLE
        Button_B(green)
        Button_L(green)
        ZbuttonGroup(green,gray,gray)
        TrigL_Sig(red)
        TrigR_Sig(green)
        #HAT EXAMPLE
        HatY(gray,green)
        HatX(red,gray)
        #JOYSTICK EXAMPLES
        JStick_1Y(40,20)
        JStick_1X(10,40)
        JStick_2Y(20,40)
        JStick_2X(40,10)
        #SENSORS EXAMPLE
        SENSOR_1_VALUE(green,0.2)
        SENSOR_2_VALUE(green,0.8)
        SENSOR_3_VALUE(red,102)
        pygame.display.update()
        pygame.display.update()
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                print("QUIT")
                running = False
                pygame.quit()



if __name__ == '__main__' :
    main()
```

BROWSER STREAMING SYSTEM PROGRAM

BROWSER SERVER PROGRAM DESCRIPTION

The program listing for this server is given in the next section. The following is an explanation of the key points in the program. The listing is annotated and the explanation refers to these annotations. The reader can use the FIND function in the Arduino IDE to refer to each annotation.

The key to this program is seen in the inclusion of `ESPAsyncWebServer.h` along with `WiFiClientSecure.h` as seen at **ANN 0**. Unlike the Python Streaming System program, these two server headers are “compatible” so small modifications to the header(s) such as required in that System, are not needed here.

ANN 1 Introduces the users’ ssid and password intro to the network.

ANN 2 introduces `Wire.h` which is the I2C library, SDA and SCL data and clock pins for I2C, arrays for data communication as described earlier in the Streaming System description chapter, and global variables for the centroid of the OpenCV color target also described in that chapter.

ANN 3 shows the pinouts for the T-Journal ESP Camera and shows the creation of the browser client port as well as the port and socket handler address for the XBOX Controller.

ANN 4 is the I2C code and data array structures needed to send and receive messages between the ESP32 and the Arduino Mega as described in the Streaming System description chapter.

ANN 5 `codePrep` is described in a special chapter entitled “The Zero Hack”.

ANN 6 shows the function which formats the data return from the server to the XBOX Controller client and again codePrep is described in “the Zero Hack” chapter.

ANN 7 show the `onWsEvent` which fires upon receipt of a socket message from the XBOX Controller and returns a message back. The latter message contains the next chapter sensor data received from the Arduino Mega via I2C as described in in the Streaming System description chapter.

ANN 8 Message parser extracts the commas from the message received from the XBOX Controller.

ANN 9 is `ExecuteCommand` function which is called from the client routine in the Arduino loop to be discussed below. This function receives messages from the browser client program discussed in the next chapter. Messages include camera’s quality, contrast, and brightness as well as messages from the OpenCV program. The OpenCV program message includes the centroid of a color tracked object (**ANN 10)** sent by an AJAX function. In addition, the `ExecuteCommand` function responds to the centroid information with a return message, **ANN 11**. This message is in

JSON format and represents a simulated sensor signal from the ESP32 relayed from the Pro Mini. This is a simulation for the future sensor installation. The JSON message is received by the AJAX program function in the browser program.

ANN 11 is the section defining the T-Journal ESP32 and Camera as well as the stream handler and the `startCameraServer` at default port 80 of the IP Address. The server then continuously transmits to the python window in the client when the client signs into the IP Address.

ANN 12 shows the setup which is fairly self-explanatory for readers with ESP Camera experience. The reader should note that near the end of the setup, the IP address is prepared for I2C transmission (to the PRO MINI as described in in the Streaming System description chapter and finally the server handler for the XBOX Controller is activated.

Unlike the Streaming System server program, the loop is not empty. It has a one-time I2C transmission of the IP Address to the Pro Mini identical to the Streaming System, but it also contains the front end of the client program which is resident in the browser. This front end will now be described; the browser program will be described in the next chapter.

ANN 13 shows the call for the `getCommand` function which can be viewed using find. The function is not easily understood unless the reader uncomments the `Serial.println(HTTP_req)` line at **ANN 14**. When this is

done, the message from the client browser can be examined in the monitor. At that point, `getCommand` becomes understandable (if the reader is willing to do a line-by-line study of it as compared to `HTTP_req`.

In summary, `HTTP_req` contains 3 message types.

The first message type, **ANN 14**, is the initial browser page load which is loaded in the absence of the `colorDetect` command, **ANN 15**. The browser page does not contain the video screen but does contain the `colorDetect` button which sends that command to the server.

When the button is pressed, the second message type, **ANN 15**, then initiates the load of browser permissions and video data. The initial video frame then appears on the browser page.

The third message type are slider commands from the browser page such as “quality”, “contrast”, and “brightness” as well as the “cm” command containing the centroid results of the color tracking found by the OpenCV routines to be described in the Browser Page chapter. Besides the centroid data a one bit Tracking is transmitted to indicate when the Tracking is “ON” or “OFF”. As mentioned above, the “cm” command initiates the transmission of an AJAX response in JSON format to the browser for display to the user. Currently the response is just

simulated digital and analog data. Actual data from sensors can be installed by an interested user in the future.

This ends the description of the SERVER for BROWSER & XBOX CONTROLLER for BROWSER program. The browser page is described in the next chapter.

BROWSER CLIENT OVERALL PROGRAM DESCRIPTION

For the most part, the program listing of this program is the same as that which was the basis of the project

INTRO TO OPENCV.JS

Readers who are unfamiliar with OpenCV may wish to follow the instructions in this chapter before proceeding. Others, or braver souls who are experienced programmers and are willing to "google as they go" can skip this section.

The reader will need an HTML editor. If the reader doesn't already have a favorite, download VS Code from https://code.visualstudio.com/Download and watch the 2-minute video at https://youtu.be/dQ31h7OgxP8.

A "lazy man" compiled version of OpenCV.js (ver. 3.4) can be downloaded from https://docs.opencv.org/3.4.0/opencv.js and put in a convenient directory on your computer. It will be called in subsequent HTML files.

If the reader is unfamiliar with OpenCV, two simple web pages which can serve as an introduction are at https://docs.opencv.org/3.4/d0/d84/tutorial_js_usage.html and https://github.com/lilyrae/opencvjs-circles.

Before using these two HTML files, generate a sample jpg file depicting a solid circular object using PAINT or other favorite application. The first HTML file simply displays the JPG but the second uses “Hough circle-detection” of an object and will draw a black circle around your circular object (HINT: Don’t generate a black object for your JPG as it will render the black outline, invisible).

BROWSER CLIENT PROGRAM DESCRIPTION

The client code is sprinkled liberally with console.log instructions which allow the user to see the results of the code. The author has used Chrome exclusively in the development of this project Chrome console.log is accessed by pressing CTRL SHIFT J simultaneously. The program listing is given in PART 2.

ANN:3 marks the address of OpenCV.js.

ANN:READY marks the Module which signals that OpenCV.js has been initialized. Once initialization is complete, the button in the left column, marked Color Detection, can be pushed (left-clicked). While faster computers do not require this capability, it is included for the sake of

completeness (the author's computer 'hangs', about 1 out of 20 times without its inclusion).

The screenshot of the client program, running on Chrome shows two columns as created by the HTML section of the code. The left column shows the original image of the camera which is transmitted at approximately 1 fps. This image, with an ID of ShowImage, is the source image of the OpenCV code routine in the program. **ANN:4** marks the creation of the src and its characteristics: rows, cols, etc.

Below the source image are the original three `imagecharacteristics` sliders in the previously referenced Santos Module and 8 new ones. In OpenCV parlance these are referred to as "trackbars". The RGB trackbars are used to set limits to the color range of colors allowed in the "processed" image in the CV application to be described in this paper. Code for the trackbars are found at **ANN:5**, **ANN:6**. The maximum and minimum values of red, green, and blue (RGB) are applied to the OpenCV function, `inRange`, at **ANN:7**. (The image is 4 channel; RGBA where A is the level of transparency. In this paper, A will set at 100% opacity, namely 255.) The code is based on the fact that besides the A Plane, the image has 3 COLOR PLANES, RGB, each pixel in each Plane having a value between 0 and 255. The high/low limits are applied to the corresponding COLOR PLANES for each pixel. Note that `inRange` has a destination image which has been created previously in the program (**ANN:8**). IMPORTANT

NOTE: EVERY IMAGE CREATED IN AN OPENCV PROGRAM HAS TO BE DELETED TO AVOID COMPUTER MEMORY LEAKAGE (ANN:8A).

The destination image mask1 is not shown in the program although it could be. However, it is used by the threshold function immediately following `inRange`. The threshold function examines the composite source image pixel value and sets the corresponding destination value at either 0 or 255 depending on whether the source value is less or greater than the threshold value. The top image in the right-hand column shows this binary image.

For the sake of completeness, an invert capability has been added to the binary image. When the INVERT button shown in the screenshot is pushed, the binary image is inverted (black becomes white, white becomes black) and subsequent processing is performed on the new image. The button is bistable so that a second push returns the binary image to its original state. This capability is not used in this paper.

In the screenshot, a red cap is the target in an ordinary room environment with an ordinary 60W fluorescent lamp. The lamp emits red, green, and blue. The red cap reflects red, green, and blue but principally red. The method of detecting the amount of each reflected color will be described now. This method allows the RGB trackbars to be set with minimal effort. Its use is strongly advised.

TARGET-COLOR PROBE

The method starts with the bottom 2 trackbars in the left column of the screenshot. These two trackbars, X and Y Probe are used to place a small white circle probe in a desired position in the bottom image of the right-hand column. The RGB values in this probe position are measured and used to set the `inRange` RGB maximums and minimums described previously. See **ANN:9,9A,9B,9C** for the code associated with this probe.

When the optimum values for a desired target are found using the X,Y Probe and set by the trackbar, the target in the binary image is white and the remainder of the image is black, ideally, as shown in the screenshot. This ideal typically can be realized only when lighting conditions can be closely controlled. Indoor, standard room lighting is acceptable. Filters can be used for optimal results but were not used here.

TRACKING

Once the binary image is deemed acceptable, the TRACKING button, which is bistable, can be clicked. **ANN:10** marks the beginning of the tracking routine. Since, as mentioned above, this paper is not concerned with the INVERT capability, only the b_invert equal to false is of interest

ANN:11 The first step in the tracking is `findContours`, which is the OpenCV algorithm which finds the contours of all the white objects in the binary image. If the tracking button is pressed when the binary image is fully black, the instructions depending on `findContours` output will throw

exceptions; the try-catch allows the program to continue safely, posting an output in the console log and the text box.

Contours is the output of `findContours` and is an array of contours of the white object(s) found in the binary image. Contours.size() finds the number of elements in the array. The hierarchy (contours inside other contours) output is not of concern here as there will be no white objects (outlined in black) inside other white objects.

ANN:12 Marks the beginning of finding the moments of the contours found. M00 is the zeroth moment-the “area” enclosed by a contour. In OpenCv it is actually the number of pixels enclosed by the contour. M10 and M01 are the x and y coordinate-weighted number of pixels enclosed. As usual, the origin of the x, y coordinate system is at the upper left corner of the image. X is positive horizontal to the right and Y is positive vertical down. Therefore M10/M00 and M01/M00 are the x, y coordinates of the centroid of a contour in the array.

ANN:13,13A marks finding the largest area contour in the array of contours using the `MaxAreaArg` function and transmitting the centroid, x_cm, y_cm to the ESP32 via a fetch instruction. During the running of the program, the centroid coordinates are seen printed in the serial monitor as well as in the console.log and in the text box in the browser screen. The ESP32 can use the centroid data for tracking purposes in robotic

applications. The author does this for the ESP32 T-JOURNAL via I2C to another subsystem as the T-JOURNAL does not have the robotic actuator capability he requires for his application. (The I2C transmission of centroid data to another subsystem is not shown here.)

ANN:14 Marks code for a blue bounding rectangle which bounds the largest-area contour and the centroid of that contour. These can be seen in the lower image in the right hand column of the browser screen, outputs of the program including the X,Y Probe data, the centroid coordinates, and a catch output if an exception is generated as mentioned above.

ANN:15 is the function which receives digital and analog data (in this case in JSON format) from the ESP32 server and sends data (in this case, target coordinates and Tracker `on_off`) back to the server.

BROWSER SYSTEM: SERVER-CLIENT CODE

ROBOT_BROWSER_SERVER_P.ino CODE

You can download the code on the following link:

- https://github.com/arshacker/ESP32-ROBOT-CAM-VIDEO-XBOX-CONTROLLER-OPENCV-SYSTEM/blob/main/ROBOT_STREAMING_SERVER_P/ROBOT_STREAMING_SERVER_P.ino

```
//ANN 1
#include "esp_camera.h"
#include <WiFi.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "soc/soc.h" //disable brownout problems
```

```
#include "soc/rtc_cntl_reg.h"  //disable brownout problems
#include "dl_lib.h"
#include "esp_http_server.h"

#include <AsyncTCP.h>
#include <ESPAsyncWebServer_ARS.h> //TOOK OUT REDEFS OF GET POST ETC IN ORIGINAL

//#define SOFTAP

/*******************FROM OCV_COLORTRACK11_MS_34_3************/
//ANN 2
#include <Wire.h>    //comm to arduino
#define myWire Wire
#define SDA 14
#define SCL 13

#define SCREENCOLS 400
#define SCREENROWS 300

#define CODE_LEN 7
int code_array[CODE_LEN] = {0,0,0,0,0,0,1};
int code_nbr = 1;

//#define MSG_LEN 7  //
#define MSG_INT_LEN 5
#define MSG_LEN  2*MSG_INT_LEN-1

//char msg[MSG_LEN] = {0,};
//int msg_int[4] = {97,98,127,1}; // {97,98,127};

//char msg[MSG_LEN] = {0,};
char msg[MSG_INT_LEN] = {0,};
int msg_int[MSG_INT_LEN] = {97,98,127,1}; // {97,98,127};
                    //ALLOW POS INTEGERS 0-127

int One_Time_Transmit = 1;

int lock = 0;
char cTT[256] = {0,};
//char c[] = {0,};
byte d[256] = {0,};
char e;

int lenGlobal=0;

byte bufIP[4] = {0,0,0,0};  //i2c xmit
byte bufToBot[20] = {0,};  //bufToBot[19] = {0,};
byte buffer[40];             //i2c rcv(prev said transmit?
byte buffer_temp[40];    //check contents before xfr to buffer
int n = 0;                   //i2c rcv

int finalIndex = 0;
int initialIndex = 0;
int kIndex = 0;

long timeStart;
long timeFinish;
long timeStart1;
long timeFinish1;
```

```
int DELTA_XCM = 0;
int DELTA_YCM = 0;
byte b_Tracker = 0;
byte b_DELTA_XCM = 0;
byte b_DELTA_YCM = 0;

//ANN 3
AsyncWebServer server3(82);

AsyncWebSocket ws("/ws");

/************************start                                                    i2c
transmit************************************/
//ANN 4
  void i2cTransmit(void){
                /************SEND I2C DATA*********************/
    //Serial.println(F("Test with 1 transmissions of writing 10 bytes each"));
  //byte buf[20] = { 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, };
  byte  buf[27]  =  {  100,  101,  102,  103,  104,  105,  106,  107,  108,
109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125,126};
  int err = 0;
  unsigned long millis1 = millis();
  boolean firsterr = false;
  //for( int i=0; i<1; i++)
  //{
    Serial.println(F("Sending data"));
    timeStart1 = millis();
    myWire.beginTransmission(4);

    for(int z=0; z<=3;z++){
      buf[z]=bufIP[z];
    }
    for(int v=0; v<=19; v++){
      buf[v+4]=bufToBot[v];
    }
    buf[24] = b_DELTA_XCM;
    buf[25] = b_DELTA_YCM;
    buf[26] = b_Tracker;

    Serial.print("BUF = ");
    for( int k=0; k<27; k++)
    {
    Serial.print( (int) buf[k]);
    Serial.print(F(", "));
    }
    Serial.println();

    myWire.write( buf, 27);       //replace with bufToBot from client
    //myWire.write(bufToBot,5);
    if( myWire.endTransmission() != 0)
    {
      err++;
    }
    delayMicroseconds(100);  // Even the normal Arduino Wire library needs some
delay when the Slave disables interrupts.
 // }

     myWire.beginTransmission(6);
```

```
     myWire.write( buf, 27);
    if( myWire.endTransmission() != 0)
    {
      err++;
    }
    delayMicroseconds(100);  // Even the normal Arduino Wire library needs some
delay when the Slave disables interrupts.


  unsigned long millis2 = millis();
  Serial.print(F("total time: "));
  Serial.print(millis2 - millis1);
  Serial.print(F(" ms, total errors: "));
  Serial.println(err);

  delay(2);

  timeFinish1 = millis();
  Serial.print("INNER TIME  = ");
  Serial.println(timeFinish1-timeStart1);

                              /**END SEND I2C DATA*****/

                              /**REQUEST I2C DATA******/

  Serial.println(F("Requesting data"));
  n = myWire.requestFrom(4, 10);    // request bytes from Slave //make n a
global
  Serial.print(F("n="));
  Serial.print(n);
  Serial.print(F(", available="));
  Serial.println(myWire.available());

//   myWire.printStatus(Serial);       // This shows information about the
SoftwareWire object.

//  byte buffer[40];         //make this a global for access by cmd func
//  for( int j=0; j<n; j++)
//    buffer[j] = myWire.read();
  myWire.readBytes( buffer_temp, n);

  Serial.print("RCV_BUFFER_TEMP = ");
  for( int k=0; k<n; k++)
  {
    if( k == 0)
      Serial.print(F("*"));      // indicate the number of the counter
    Serial.print( (int) buffer_temp[k]);
    Serial.print(F(", "));
  }
  Serial.println();

  if(buffer_temp[0]<128){                //NO ERROR
    for( int k=0; k<n; k++){
      buffer[k] = buffer_temp[k];
    }
  }

  Serial.print("RCV_BUFFER = ");
  for( int k=0; k<n; k++)
```

```
  {
    if( k == 0)
      Serial.print(F("*"));      // indicate the number of the counter
    Serial.print( (int) buffer[k]);
    Serial.print(F(", "));
  }
  Serial.println();

                                  /***END              REQUEST              I2C
DATA*********/
  delay(2);

  }

 /***********************end i2c transmit*********************************/
//ANN 5
void codePrep(void){
  code_nbr = 2*2*2*2*2*2*code_array[0] + 2*2*2*2*2*code_array[1] +
             2*2*2*2*code_array[2] + 2*2*2*code_array[3]+
             2*2*code_array[4] + 2*code_array[5] + code_array[6];
  Serial.print("CODE NBR = ");
  Serial.println(code_nbr);
}

//ANN 6
void msgPrep(void){
  //takes msg_int[3]={X,Y,Z] and creates msg[X,',',Y,',',Z} WHERE MSG_LEN = 5
/*
  for(int i = 1; i < MSG_LEN-2; i = i + 1){
    msg[(2*i)-1] = ',';  //comma separators for python
                         //msg[1],msg[3]...
  }
  for(int j = 1; j < MSG_LEN-1; j = j + 1){
    if(buffer[(j-1)]!=0){
        msg[(2*j)-2] = buffer[(j-1)];
    }
    else{
        msg[(2*j)-2] = 1;     //cant transmit 0 to python?????
    }
    //msg[(2*j)-2] = msg_int[(j-1)]; //prepared message
      //msg[0]=msg_int[0],msg[2]=msg_int[1],msg[4]=msg_int[2]...
  }
 */
 //below is most recent before new strategy
 /*
  for(int i = 1; i<MSG_INT_LEN+1; i = i + 1){
    msg[(2*i)-1] = 2;  //MARKER 2 SAYS IT IS NOT ZERO
  }
  for(int i = 0; i<MSG_INT_LEN; i = i + 1){
    if (buffer[i]!=0){
      msg[2*i]=buffer[i];
    }
    else{
      msg[2*i]=1;  //cant transmit 0 (null) to python
      msg[(2*i)+1]=3;  //MARKER 3 SAYS IT SHOULD BE ZERO
    }
  }
  */
```

```
  for(int j=0; j<CODE_LEN-1; j=j+1){   //leave last location as 1
    code_array[j] = 0;                 //code_array(CODE_LEN-1)==1
  }

  for(int i = 0; i<MSG_INT_LEN; i = i + 1){
    if (buffer[i]!=0){
      msg[i]= buffer[i];
      code_array[i] = 0;
    }
    else{
      msg[i] = 1;  //cant transmit 0 (null) to python
      code_array[i] = 1;
    }
  }

  codePrep();
  msg[MSG_INT_LEN-1] = (char)code_nbr; //put code into MSG_INT_LEN location

  //Serial.print("BUFFER BUFFERR BUFFER ===");
  //Serial.println(buffer[1]); //prints a
  //msg[2] = '\x01';
  //msg[1] = 5;
  //msg[0] = buffer[0];
  //msg[2] = buffer[1];
  //msg[4] = buffer[2];
  //msg[6] = buffer[3];
}

//ANN 7
void  onWsEvent(AsyncWebSocket  *  server3,  AsyncWebSocketClient  *  client,
AwsEventType type, void * arg, uint8_t *data, size_t len){

  //msgPrep(); //comma separators  //put in ws_evt_data

  //int msg_int[] = {97,',',98,',',127}; //commas separators useful in python
  //char msg[5] = {0,};
  //char msg[] = {'a','b','c'};
  //char msg[] = {97,98,99};
  //char msg[] = {97,',',98,',',99};
  //char msg[] = {97,',',98,',',31};

  //msg[0] = msg_int[0]; //int to char for transmission
  //msg[1] = msg_int[1];
  //msg[2] = msg_int[2];
  //msg[3] = msg_int[3];
  //msg[4] = msg_int[4];

  lenGlobal = len;

  if(type == WS_EVT_CONNECT){

    Serial.println("Websocket client connection received");
    client->text("Hello from ESP32 Server3");
    //client->binary(msg); //move to ws_evt_data

  } else if(type == WS_EVT_DISCONNECT){
    Serial.println("Client disconnected");

  }else if(type==WS_EVT_DATA){
```

```
    Serial.println("Data received: ");

    for(int j=0; j < len; j++){
      d[j] = data[j];          //data read out only once allowed
      cTT[j] = d[j];  //ascii to char


      Serial.print(cTT[j]);   //TIMING
    }//end for j loop
    Serial.println();
    Serial.println(len);



    for(int i=0; i < len; i++){
      Serial.print(d[i]);     //TIMING
      Serial.print("|");
    }
    Serial.println();

    parse_msg();           //convert msg from python to array format
    initializeTT();


    msgPrep(); //comma separators
    client->binary(msg);  //return msg to python

    Serial.println("I2C Transmit");
    i2cTransmit();
    initializeTT();

   }//end else if ws evt data
}//end onwsevent


void initializeTT(void)
{
  for(int n=0;n<256;n++){
    cTT[n] = 0;
    d[n] = 0;
   }
}

/*
void initialize(void)
{
  for(int n=0;n<255;n++)
  {c[n] = 0;}
}
*/

//ANN 8
void parse_msg(void){

   Serial.print("cTT = ");
   for(int w=0; w<lenGlobal;w=w+1){
      Serial.print(cTT[w]);
   }
   Serial.println();
```

```
   String var = String(cTT);
   var = String('[') + var + String(']');
   Serial.print("var = ");
   Serial.println(var);

    /*******************parse on comma*********************************/
  /*******************orig parse**************************
  int index1 = var.indexOf(',',0);
  int index2 = var.indexOf(',',index1+1);
  int index3 = var.indexOf(',',index2+1);
  int index4 = var.indexOf(',',index3+1);
  int index5 = var.indexOf(',',index4+1);

  int indexEND = var.indexOf('?');  //will return -1
  Serial.println(indexEND);
  String X_VALUE = var.substring(1,index1); //omit the [
  String Y_VALUE = var.substring(index1+1,index2);
  String A_VALUE = var.substring(index2+1,index3);
  String B_VALUE = var.substring(index3+1,index4);
  String C_VALUE = var.substring(index4+1,index5);
  Serial.println(X_VALUE);
  Serial.println(Y_VALUE);
  Serial.println(C_VALUE);
  int x_value_int = X_VALUE.toInt();
  x_value_int++;
  Serial.println(x_value_int);
  bufToBot[0] = X_VALUE.toInt();
  bufToBot[1] = Y_VALUE.toInt();
  bufToBot[2] = A_VALUE.toInt();
  bufToBot[3] = B_VALUE.toInt();
  //bufToBot[4] = C_VALUE.toInt(); //SEND COUNT TO BOT only goes to 255,then
starts again
  ************************end orig parse****************************/

  /****************************new parse*******************************/
  initialIndex = 1;  //IGNORE THIS start initialIndex at 1 b/c python transmits
[ .
  kIndex = 0;
  finalIndex = 0;
  while(finalIndex!=-1){
    finalIndex = var.indexOf(',',initialIndex);
    bufToBot[kIndex] = var.substring(initialIndex,finalIndex).toInt();
    //Serial.println(bufToBot[kIndex]);
    if(finalIndex==-1){
      //Serial.println(bufToBot[kIndex]);
      initialIndex = 0;
      finalIndex = 0;
      kIndex = 0;
      break;}
    initialIndex = finalIndex+1;
    kIndex++;
  }
/************************end new parse*************************************/
  Serial.print("BUFTOBOT = ");
  for(int m=0;m<20;m++){      //NEW
    Serial.print(bufToBot[m]);
    Serial.print(" ");
  }
```

```
  Serial.println();
    /*********************end parse on comma*****************************/
}

/*********************END FROM OCV_COLORTRACK11_MS_34_3******************/

//Replace with your network credentials

//ANN 10
const char* ssid = "ssid";
const                    char*                    password                    =
"password";

const char* ssidAP = "ssidAP"; // ars  soft AP
const char* passwordAP = "passwordAP"; //ars softAP


#define PART_BOUNDARY "123456789000000000000987654321"

// This project was tested with the AI Thinker Model, M5STACK PSRAM Model and
M5STACK WITHOUT PSRAM
#define CAMERA_MODEL_T_JOURNAL
//#define CAMERA_MODEL_AI_THINKER
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WITHOUT_PSRAM

// Not tested with this model
//#define CAMERA_MODEL_WROVER_KIT

/*#if defined(CAMERA_MODEL_WROVER_KIT)
  #define PWDN_GPIO_NUM    -1
  #define RESET_GPIO_NUM   -1
  #define XCLK_GPIO_NUM    21
  #define SIOD_GPIO_NUM    26
  #define SIOC_GPIO_NUM    27

  #define Y9_GPIO_NUM      35
  #define Y8_GPIO_NUM      34
  #define Y7_GPIO_NUM      39
  #define Y6_GPIO_NUM      36
  #define Y5_GPIO_NUM      19
  #define Y4_GPIO_NUM      18
  #define Y3_GPIO_NUM       5
  #define Y2_GPIO_NUM       4
  #define VSYNC_GPIO_NUM   25
  #define HREF_GPIO_NUM    23
  #define PCLK_GPIO_NUM    22
#elif defined(CAMERA_MODEL_M5STACK_PSRAM)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     25
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
```

```
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       32
  #define VSYNC_GPIO_NUM    22
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21
#elif defined(CAMERA_MODEL_M5STACK_WITHOUT_PSRAM)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     25
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       17
  #define VSYNC_GPIO_NUM    22
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21
#elif defined(CAMERA_MODEL_AI_THINKER)
  #define PWDN_GPIO_NUM     32
  #define RESET_GPIO_NUM    -1
  #define XCLK_GPIO_NUM      0
  #define SIOD_GPIO_NUM     26
  #define SIOC_GPIO_NUM     27

  #define Y9_GPIO_NUM       35
  #define Y8_GPIO_NUM       34
  #define Y7_GPIO_NUM       39
  #define Y6_GPIO_NUM       36
  #define Y5_GPIO_NUM       21
  #define Y4_GPIO_NUM       19
  #define Y3_GPIO_NUM       18
  #define Y2_GPIO_NUM        5
  #define VSYNC_GPIO_NUM    25
  #define HREF_GPIO_NUM     23
  #define PCLK_GPIO_NUM     22
#elif defined(CAMERA_MODEL_T_JOURNAL)*/

//ANN 11
//       T-JOURNAL                    AI_THINKER
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM     27     //         0
#define SIOD_GPIO_NUM     25     //        26
#define SIOC_GPIO_NUM     23     //        27

#define Y9_GPIO_NUM       19     //        35
#define Y8_GPIO_NUM       36     //        34
#define Y7_GPIO_NUM       18     //        39
#define Y6_GPIO_NUM       39     //        36
#define Y5_GPIO_NUM        5     //        21
#define Y4_GPIO_NUM       34     //        19
#define Y3_GPIO_NUM       35     //        18
```

```
#define Y2_GPIO_NUM       17     //         5
#define VSYNC_GPIO_NUM    22     //        25
#define HREF_GPIO_NUM     26     //        23
#define PCLK_GPIO_NUM     21     //        22
/*#else
  #error "Camera model not selected"
#endif*/

//ANN 12
static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary="
PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length:
%u\r\n\r\n";

httpd_handle_t stream_httpd = NULL;

static esp_err_t stream_handler(httpd_req_t *req){
  camera_fb_t * fb = NULL;
  esp_err_t res = ESP_OK;
  size_t _jpg_buf_len = 0;
  uint8_t * _jpg_buf = NULL;
  char * part_buf[64];

  res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
  if(res != ESP_OK){
    return res;
  }

  while(true){
    fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Camera capture failed");
      res = ESP_FAIL;
    } else {
      if(fb->width > 400){
        if(fb->format != PIXFORMAT_JPEG){
          bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
          esp_camera_fb_return(fb);
          fb = NULL;
          if(!jpeg_converted){
            Serial.println("JPEG compression failed");
            res = ESP_FAIL;
          }
        } else {
          _jpg_buf_len = fb->len;
          _jpg_buf = fb->buf;
        }
      }
    }
    if(res == ESP_OK){
      size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
      res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
    }
    if(res == ESP_OK){
```

```
      res         =         httpd_resp_send_chunk(req,         _STREAM_BOUNDARY,
strlen(_STREAM_BOUNDARY));
    }
    if(fb){
      esp_camera_fb_return(fb);
      fb = NULL;
      _jpg_buf = NULL;
    } else if(_jpg_buf){
      free(_jpg_buf);
      _jpg_buf = NULL;
    }
    if(res != ESP_OK){
      break;
    }
    //Serial.printf("MJPG: %uB\n",(uint32_t)(_jpg_buf_len));
  }
  return res;
}

void startCameraServer(){
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  config.server_port = 80;

  httpd_uri_t index_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
  };

  //Serial.printf("Starting web server on port: '%d'\n", config.server_port);
  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &index_uri);
  }
}

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector

  Serial.begin(115200);
  Serial.setDebugOutput(false);
  Serial.println();

  myWire.begin(SDA,SCL);

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
```

```
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;

  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

  // Camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
 // Wi-Fi connection

 #if defined(SOFTAP)
/*****************SOFTAP**********************************/
  WiFi.softAP(ssidAP,NULL,1,0,1);
  // ssid,pwd,channel(1-13), broadcast/hidden, max connections(4)
  Serial.print("Setting AP..");
  IPAddress ip = WiFi.softAPIP();

  /*
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  */
  Serial.print("softIP = ");
  Serial.println(ip);

  Serial.println("");
  Serial.println("WiFi connected");

  Serial.print("Camera Stream Ready! Go to: http://");
  //Serial.println(WiFi.localIP()); ars using softAP
  //Serial.println(IP);
/**********************END SOFTAP*****************************/


 #else
 /***************STATION POINT**********************/
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
```

```
  Serial.println("WiFi connected");

  Serial.print("Camera Stream Ready! Go to: http://");
  Serial.print(WiFi.localIP());
  IPAddress ip = WiFi.localIP();
   /*******************END STATION POINT*************************/

 #endif

  /*******************FROM OCV_COLORTRACK11_MS_34_3************/
    /*******************transmit ip address*****************************/


    //bufIP[0] = ip[0];
  //Serial.println(String(ip[0]));
  String ipString = String(ip[0])+String('.')+String(ip[1])+String('.')+
                       String(ip[2])+String('.')+String(ip[3]);
  Serial.println(ipString);
  int z = String(ip[0]).toInt();
  z=z+1;
  Serial.println(z);
  bufIP[0] = (byte)String(ip[0]).toInt();  //bufIP[] is byte array
  Serial.println(bufIP[0]);
  for(int k=0; k<=3; k++){
    //bufIP[k] = (byte)String(ip[k]).toInt();  //bufIP[] is byte array
    bufIP[k] = String(ip[k]).toInt();  //bufIP[] is byte array
  }
  //Serial.println((int)bufIP[2]);
  Serial.println(bufIP[1]);
  Serial.println(bufIP[2]);
  Serial.println(bufIP[3]);

  /***********************end                    transmit                    ip
address***********************************/


  ws.onEvent(onWsEvent);
  server3.addHandler(&ws);

  server3.begin();
  /*******************END FROM OCV_COLORTRACK11_MS_34_3************/

  // Start streaming web server
  startCameraServer();
}
//ANN 13
void loop() {
  delay(1);

  /*******************FROM OCV_COLORTRACK11_MS_34_3************/
  if(One_Time_Transmit==1){
     delay(4000);
     initializeTT();
     i2cTransmit();
     One_Time_Transmit = 0;
  }
   /*******************END FROM OCV_COLORTRACK11_MS_34_3************/
}
```

index_OCV_ColorTrack.h CODE

You can download the code on the following link:

- https://github.com/arshacker/ESP32-ROBOT-CAM-BROWSER-XBOX-CONTROLLER-OPENCV-SYSTEM/blob/main/ROBOT_BROWSER_SERVER_P/index_OCV_ColorTrack.h

---

```
/*****************************
  This include file, index_OCV_ColorTrack.h, the Client, is an intoduction of
OpenCV.js to the ESP32 Camera environment. The Client was
  developed and written by Andrew R. Sass. Permission to reproduce the
index_OCV_ColorTrack.h file is granted free of charge if this
  entire copyright notice is included in all copies of the
index_OCV_ColorTrack.h file.
*****************************/
static const char PROGMEM INDEX_HTML[] = R"rawliteral(
<!DOCTYPE html>
<head>
   <title>ESP32-CAMERA COLOR DETECTION</title>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width,initial-scale=1">
   <!----ANN:3--->
   <!--<script          async          src="http://192.168.4.2:8080/opencv.js"
type="text/javascript"></script>-->
   <script      async      src="      https://docs.opencv.org/master/opencv.js"
type="text/javascript"></script>
</head>
<style>
  body { background-color: #808080;}
  .column{
    float: left;
    width: 50%
  }
  .row:after{
    content: "";
    display: table;
    clear: both;
  }
</style>
<body>
<div class="container">
  <div class = "row">
    <div class = "column">
      <img id="ShowImage" src="" style="display:none">
      <canvas id="canvas" style="display:none"></canvas>

      <table>
      <tr>
      <td   colspan="2"><input   type="button"   id="colorDetect"   value="Color
Detection" style="display:none"></td>
      <td><input type="button" id="restart" value="Reset Board"></td>
      </tr>
      <tr>
```

```
        <td>TeleOp</td>
        <td colspan="2">
          <select id="mirrorimage">
            <option value="1">yes</option>
            <option value="0">no</option>
          </select>
        </td>
        <!--<td colspan="6">-->
        <!--<td>SENSOR1</td>-->
        <!--<td colspan="2">-->
        <!--<td  colspan="2"><input  type="slideOutput"  id="SENSOR1"  min="0"
max="127" value="40"></td>-->
        <!--</td>-->
        <td>
        <canvas  id="textCanvas0"  width="400"  height="80"  style=  "border:  1px
solid #black;"></canvas>
        </td>
      </tr>
      <tr>
      <td>Quality</td>
      <td  colspan="2"><input  type="range"  id="quality"  min="10"  max="63"
value="10"></td>
      </tr>
      <tr>
      <td>Brightness</td>
      <td  colspan="2"><input  type="range"  id="brightness"  min="-2"  max="2"
value="0"></td>
      </tr>
      <tr>
      <td>Contrast</td>
      <td  colspan="2"><input  type="range"  id="contrast"  min="-2"  max="2"
value="0"></td>
      </tr>
      </table>

      <!-----ANN:5---->
      <div>
        <p>RGB COLOR TRACKBARS
      </div>
      <div class = "slidecontainer">
          <input type="range" id="rmin" min="0" max="255"
              value="0" class = "slider">
          <input type="range" id="rmax" min="0" max="255"
              value="50" class = "slider">
              <p>RMIN:<span id="RMINdemo"></span>
                                 
              RMAX:<span id="RMAXdemo"></span></p>
      </div>
      <div class = "slidecontainer">
          <input type="range" id="gmin" min="0" max="255"
              value="0" class = "slider">
          <input type="range" id="gmax" min="0" max="255"
             value="50" class = "slider">
          <p>GMIN:<span id="GMINdemo"></span>
                                
             GMAX:<span id="GMAXdemo"></span></p>
      </div>
      <div class = "slidecontainer">
          <input type="range" id="bmin" min="0" max="255"
```

```
            value="0" class = "slider">
        <input type="range" id="bmax" min="0" max="255"
            value="50" class = "slider">
        <p>BMIN:<span id="BMINdemo"></span>
                              
           BMAX:<span id="BMAXdemo"></span></p>
    </div>
    <div>
      <p>THRESHHOLD MINIMUM-BINARY IMAGE
    </div>
    <div class = "slidecontainer">
        <input type="range" id="thresh_min" min="0" max="255"
            value="120" class = "slider">
        <p>THRESH_MIN:<span id="THRESH_MINdemo"></span>
    </div>
   <!----ANN:9--->
    <div>
      <p>COLOR PROBE
    </div>
    <div class = "slidecontainer">
        <input type="range" id="x_probe" min="0" max="400"
            value="200" class = "slider">
        <input type="range" id="y_probe" min="0" max="296"
            value="148"                          class                          =
"slider">
        <p>X_PROBE:<span id="X_PROBEdemo"></span>
                              
           Y_PROBE:<span id="Y_PROBEdemo"></span>
                              
           AJAX_INPUT:<span id="sw_an_data"></span></p>
    </div>

  </div>   <!------endfirstcolumn---------------->

  <div class = "column">
    <div>
    Image Mask
    </div>
    <div>
    <canvas id="imageMask"></canvas>
    </div>
    <div>
    Image Canvas
    </div>
    <div>
    <canvas id="imageCanvas"></canvas>
    </div>
    <div class="list-group-item">
      <button     type="button"     id="invertButton"     class="btn     btn-
primary">INVERT</button>
        <p>INVERT:<span id="INVERTdemo"></span></p>
    </div>
    <div class="list-group-item">
      <button type="button" id="contourButton" class="btn btn-primary">SHOW
CONTOUR</button>
               
      <button     type="button"     id="trackButton"     class="btn     btn-
primary">TRACKING</button>
```

```
          <p>CONTOUR:<span  id="CONTOURdemo"></span>          
          
             TRACK:<span  id="TRACKdemo"></span>             
 
             XCM:<span id="XCMdemo"></span>           
             YCM:<span  id="YCMdemo"></span>             
 </p>
      </div>
      <div>
      <canvas  id="textCanvas"  width="480"  height="180"  style=  "border:  1px
solid #black;"></canvas>
      </div>
      <iframe id="ifr" style="display:none"></iframe>
      <div id="message" style="color:green"><div>
    </div>  <!------end2ndcolumn---------------------->
  </div>   <!-----endrow---------------------->
</div>   <!------endcontainer--------------->

 <!--------------- </body>----------------->
 <!----------------</html>----------------->
<div class="modal"></div>
<script>
var sensor1 = document.getElementById("SENSOR1");
var colorDetect = document.getElementById('colorDetect');
var ShowImage = document.getElementById('ShowImage');
var canvas = document.getElementById("canvas");
var context = canvas.getContext("2d");
var mirrorimage = document.getElementById("mirrorimage");
var imageMask = document.getElementById("imageMask");
var imageMaskContext = imageMask.getContext("2d");
var imageCanvas = document.getElementById("imageCanvas");
var imageContext = imageCanvas.getContext("2d");
var txtcanvas = document.getElementById("textCanvas");
var txtcanvas0 = document.getElementById("textCanvas0");
var ctx = txtcanvas.getContext("2d");
var ctx0 = txtcanvas0.getContext("2d");
var message = document.getElementById('message');
var ifr = document.getElementById('ifr');
var myTimer;
var restartCount=0;
const modelPath = 'https://ruisantosdotme.github.io/face-api.js/weights/';
let currentStream;
let displaySize = { width:400, height: 296 }
let faceDetection;
var ADCarr = 0;
var ans = [10,20];
var asensor = 20;
var bsensor = 0;
var ana_sensor1 = 0;
var ana_sensor2 = 0;
var dig_sensor1 = 0;
var dig_sensor2 = 0;
let b_tracker = false;
let b_tracker_int = 0;
let x_cm = 0;
let y_cm = 0;
var json_obj;
let b_invert = false;
let b_contour = false;
```

```
var RMAX=50;
var RMIN=0;
var GMAX=50;
var GMIN=0;
var BMAX=50;
var BMIN=0;
var THRESH_MIN=120;
var X_PROBE=200;
var Y_PROBE=196;
var R=0;
var G=0;
var B=0;
var A=0;
colorDetect.onclick = function (event) {
  clearInterval(myTimer);
  myTimer = setInterval(function(){error_handle();},5000);
  ShowImage.src=location.origin+'/?colorDetect='+Math.random();
  //setInterval('GetSwitchAnalogData()', 1000);
}
//ANN:READY
var Module = {
  onRuntimeInitialized(){onOpenCvReady();}
}
function onOpenCvReady(){
  //alert("onOpenCvReady");
  console.log("OpenCV IS READY!!!");
  drawReadyText();
  document.body.classList.remove("loading");
}

function error_handle() {
  restartCount++;
  clearInterval(myTimer);
  if (restartCount<=2) {
    message.innerHTML    =    "Get    still    error.    <br>Restart    ESP32-CAM
"+restartCount+" times.";
    myTimer = setInterval(function(){colorDetect.click();},10000);
    ifr.src = document.location.origin+'?restart';
  }
  else
    message.innerHTML = "Get still error. <br>Please close the page and check
ESP32-CAM.";
}
colorDetect.style.display = "block";
ShowImage.onload = function (event) {
  //alert("SHOW IMAGE");
  console.log("SHOW iMAGE");
  clearInterval(myTimer);
  restartCount=0;
  canvas.setAttribute("width", ShowImage.width);
  canvas.setAttribute("height", ShowImage.height);
  canvas.style.display = "block";
  if (mirrorimage.value==0) {
    //context.translate((canvas.width + ShowImage.width) / 2, 0);
    //context.scale(-1, 1);
    //context.drawImage(ShowImage, 0, 0, ShowImage.width, ShowImage.height);
    //context.drawImage(ShowImage,       400,       400,       ShowImage.width,
ShowImage.height);
    context.drawImage(ShowImage, 0, 0, ShowImage.width, ShowImage.height);
```

```
    //context.drawImage(ShowImage, 0, 0, 20,20);//SUPER SMALL
    //context.setTransform(1, 0, 0, 1, 0, 0);
    imageCanvas.setAttribute("width", ShowImage.width);
    imageCanvas.setAttribute("height", ShowImage.height);
    imageCanvas.style.display = "block";
    imageMask.setAttribute("width", ShowImage.width);
    imageMask.setAttribute("height", ShowImage.height);
    imageMask.style.display = "block";
  }
  else {
    //context.drawImage(ShowImage,0,0,ShowImage.width,ShowImage.height);
    canvas.setAttribute("width", 1600);
    canvas.setAttribute("height", 800);
    context.drawImage(ShowImage,0,0,1600,800);
    imageCanvas.setAttribute("width", 0);
    imageCanvas.setAttribute("height", 0);
    imageCanvas.style.display = "block";
    imageMask.setAttribute("width", 0);
    imageMask.setAttribute("height", 0);
    imageMask.style.display = "block";

  }

  DetectImage();
}
restart.onclick = function (event) {
  fetch(location.origin+'/?restart=stop');
}
quality.onclick = function (event) {
  fetch(document.location.origin+'/?quality='+this.value+';stop');
}
brightness.onclick = function (event) {
  fetch(document.location.origin+'/?brightness='+this.value+';stop');
}
contrast.onclick = function (event) {
  fetch(document.location.origin+'/?contrast='+this.value+';stop');
}
async function DetectImage() {
  //alert("DETECT IMAGE");
  console.log("DETECT IMAGE");
GetSwitchAnalogData();
asensor = asensor + 1;
  /***************opencv*********************************/
  //ANN:4
  let src = cv.imread(ShowImage);
  arows = src.rows;
  acols = src.cols;
  aarea = arows*acols;
  adepth = src.depth();
  atype = src.type();
  achannels = src.channels();
  console.log("rows = " + arows);
  console.log("cols = " + acols);
  console.log("pic area = " + aarea);
  console.log("depth = " + adepth);
  console.log("type = " + atype);
  console.log("channels = " + achannels);
  console.log("ASENSOR = " + asensor);
  console.log("BSENSOR = " + bsensor);
```

```
/******************COLOR DETECT*****************************/
//ANN:6
var RMAXslider = document.getElementById("rmax");
var RMAXoutput = document.getElementById("RMAXdemo");
RMAXoutput.innerHTML = RMAXslider.value;
RMAXslider.oninput = function(){
RMAXoutput.innerHTML = this.value;
RMAX = parseInt(RMAXoutput.innerHTML,10);
console.log("RMAX=" + RMAX);
}
console.log("RMAX=" + RMAX);
var RMINslider = document.getElementById("rmin");
var RMINoutput = document.getElementById("RMINdemo");
RMINoutput.innerHTML = RMINslider.value;
RMINslider.oninput = function(){
  RMINoutput.innerHTML = this.value;
  RMIN = parseInt(RMINoutput.innerHTML,10);
  console.log("RMIN=" + RMIN);
}
console.log("RMIN=" + RMIN);
var GMAXslider = document.getElementById("gmax");
var GMAXoutput = document.getElementById("GMAXdemo");
GMAXoutput.innerHTML = GMAXslider.value;
GMAXslider.oninput = function(){
  GMAXoutput.innerHTML = this.value;
  GMAX = parseInt(GMAXoutput.innerHTML,10);
}
console.log("GMAX=" + GMAX);
var GMINslider = document.getElementById("gmin");
var GMINoutput = document.getElementById("GMINdemo");
GMINoutput.innerHTML = GMINslider.value;
GMINslider.oninput = function(){
  GMINoutput.innerHTML = this.value;
  GMIN = parseInt(GMINoutput.innerHTML,10);
}
console.log("GMIN=" + GMIN);
var BMAXslider = document.getElementById("bmax");
var BMAXoutput = document.getElementById("BMAXdemo");
BMAXoutput.innerHTML = BMAXslider.value;
BMAXslider.oninput = function(){
  BMAXoutput.innerHTML = this.value;
  BMAX = parseInt(BMAXoutput.innerHTML,10);
}
console.log("BMAX=" + BMAX);
var BMINslider = document.getElementById("bmin");
var BMINoutput = document.getElementById("BMINdemo");
BMINoutput.innerHTML = BMINslider.value;
BMINslider.oninput = function(){
BMINoutput.innerHTML = this.value;
BMIN = parseInt(BMINoutput.innerHTML,10);
}
console.log("BMIN=" + BMIN);
var THRESH_MINslider = document.getElementById("thresh_min");
var THRESH_MINoutput = document.getElementById("THRESH_MINdemo");
THRESH_MINoutput.innerHTML = THRESH_MINslider.value;
THRESH_MINslider.oninput = function(){
THRESH_MINoutput.innerHTML = this.value;
THRESH_MIN = parseInt(THRESH_MINoutput.innerHTML,10);
}
```

```
  console.log("THRESHOLD MIN=" + THRESH_MIN);
  //ANN:9A
  var X_PROBEslider = document.getElementById("x_probe");
  var X_PROBEoutput = document.getElementById("X_PROBEdemo");
  X_PROBEoutput.innerHTML = X_PROBEslider.value;
  X_PROBEslider.oninput = function(){
  X_PROBEoutput.innerHTML = this.value;
  X_PROBE = parseInt(X_PROBEoutput.innerHTML,10);
  }
  console.log("X_PROBE=" + X_PROBE);
  var Y_PROBEslider = document.getElementById("y_probe");
  var Y_PROBEoutput = document.getElementById("Y_PROBEdemo");
  Y_PROBEoutput.innerHTML = Y_PROBEslider.value;
  Y_PROBEslider.oninput = function(){
  Y_PROBEoutput.innerHTML = this.value;
  Y_PROBE = parseInt(Y_PROBEoutput.innerHTML,10);
  }
  console.log("Y_PROBE=" + Y_PROBE);
  var ZETA = document.getElementById("sw_an_data");
  //ZETA.innerHTML = "ON"; //TEST
  document.getElementById('trackButton').onclick = function(){
    b_tracker = (true && !b_tracker)
    if(b_tracker==true){
      b_tracker_int = 1;
    }
    if(b_tracker==false){
      b_tracker_int = 0;
    }
    console.log("TRACKER = " + b_tracker );
    console.log("TRACKER INT = " + b_tracker_int );
    var TRACKoutput = document.getElementById("TRACKdemo");
    TRACKoutput.innerHTML = b_tracker;
    //var XCMoutput = document.getElementById("XCMdemo");
    //XCMoutput.innerHTML = x_cm;

  }
  document.getElementById('invertButton').onclick = function(){
    b_invert = (true && !b_invert)
    console.log("TRACKER = " + b_invert );
    var INVERToutput = document.getElementById("INVERTdemo");
    INVERToutput.innerHTML = b_invert;
  }
/**/
  document.getElementById('contourButton').onclick = function(){
    b_contour = (true && !b_contour)
    console.log("TRACKER = " + b_contour );
    var CONTOURoutput = document.getElementById("CONTOURdemo");
    CONTOURoutput.innerHTML = b_contour;
  }
/**/
  let tracker = 0;

  var TRACKoutput = document.getElementById("TRACKdemo");
  TRACKoutput.innerHTML = b_tracker;
  var XCMoutput = document.getElementById("XCMdemo");
  var YCMoutput = document.getElementById("YCMdemo");
  XCMoutput.innerHTML = 0;
  YCMoutput.innerHTML = 0;
  var INVERToutput = document.getElementById("INVERTdemo");
```

```
INVERToutput.innerHTML = b_invert;
var CONTOURoutput = document.getElementById("CONTOURdemo");
CONTOURoutput.innerHTML = b_contour;
//ANN:8
let M00Array = [0,];
let orig = new cv.Mat();
let mask = new cv.Mat();
let mask1 = new cv.Mat();
let mask2 = new cv.Mat();
let contours = new cv.MatVector();
let hierarchy = new cv.Mat();
let rgbaPlanes = new cv.MatVector();

let color = new cv.Scalar(0,0,0);
clear_canvas();
clear_canvas0();

orig = cv.imread(ShowImage);
cv.split(orig,rgbaPlanes);  //SPLIT
let BP = rgbaPlanes.get(2);  // SELECTED COLOR PLANE
let GP = rgbaPlanes.get(1);
let RP = rgbaPlanes.get(0);
cv.merge(rgbaPlanes,orig);


            //   BLK    BLU   GRN   RED
let row = Y_PROBE //180//275 //225 //150 //130
let col = X_PROBE //100//10 //100 //200 //300
drawColRowText(acols,arows);
//drawASensorText(asensor,bsensor);
drawASensorText(ana_sensor1,dig_sensor1,ana_sensor2,dig_sensor2);
console.log("ISCONTINUOUS = " + orig.isContinuous());
//console.log("TRACKER = " + b_tracker );
console.log("TRACKER INT = " + b_tracker_int );
//ANN:9C
R = src.data[row * src.cols * src.channels() + col * src.channels()];
G = src.data[row * src.cols * src.channels() + col * src.channels() + 1];
B = src.data[row * src.cols * src.channels() + col * src.channels() + 2];
A = src.data[row * src.cols * src.channels() + col * src.channels() + 3];
console.log("RDATA = " + R);
console.log("GDATA = " + G);
console.log("BDATA = " + B);
console.log("ADATA = " + A);
drawRGB_PROBE_Text();



//ANN:9b
//*************draw probe point********************
let point4 = new cv.Point(col,row);
cv.circle(src,point4,5,[255,255,255,255],2,cv.LINE_AA,0);
//***********end draw probe point*********************
//ANN:7
let high = new cv.Mat(src.rows,src.cols,src.type(),[RMAX,GMAX,BMAX,255]);
let low = new cv.Mat(src.rows,src.cols,src.type(),[RMIN,GMIN,BMIN,0]);
cv.inRange(src,low,high,mask1);
//inRange(source image, lower limit, higher limit, destination image)

cv.threshold(mask1,mask,THRESH_MIN,255,cv.THRESH_BINARY);
```

```
  //threshold(source image,destination image,threshold,255,threshold method);
  //ANN:9
  if(b_invert==true){
     cv.bitwise_not(mask,mask2);
  }

  //dataTransmit2(5);

  //GetSwitchAnalogData();
  console.log("ADCARR = " + ADCarr);



/********************start contours*****************************************/
  //ANN:10
  if(b_tracker == true){
  try{
   if(b_invert==false){
    //ANN:11
    cv.findContours(mask,contours,hierarchy,cv.RETR_CCOMP,cv.CHAIN_APPROX_SIM
PLE);
    //findContours(source image, array of contours found, hierarchy of contours
        // if contours are inside other contours, method of contour data
retrieval,
        //algorithm method)
   }
   else{
    cv.findContours(mask2,contours,hierarchy,cv.RETR_CCOMP,cv.CHAIN_APPROX_SI
MPLE);
   }
    console.log("CONTOUR_SIZE = " + contours.size());
    //draw contours
    if(b_contour==true){
     for(let i = 0; i < contours.size(); i++){
        cv.drawContours(src,contours,i,[0,0,0,255],2,cv.LINE_8,hierarchy,100)
     }
    }
    //ANN:12
    let cnt;
    let Moments;
    let M00;
    let M10;
    //let x_cm;
    //let y_cm;

    //ANN:13
    for(let k = 0; k < contours.size(); k++){
        cnt = contours.get(k);
        Moments = cv.moments(cnt,false);
        M00Array[k] = Moments.m00;
       // cnt.delete();
    }
    //ANN13A
    let max_area_arg = MaxAreaArg(M00Array);
    console.log("MAXAREAARG = "+max_area_arg);
    //let TestArray = [0,0,0,15,4,15,2];
    //let TestArray0 = [];
    //let max_test_area_arg = MaxAreaArg(TestArray0);
    //console.log("MAXTESTAREAARG = "+max_test_area_arg);
```

```
    let ArgMaxArea = MaxAreaArg(M00Array);
    if(ArgMaxArea >= 0){
    cnt = contours.get(MaxAreaArg(M00Array));  //use the contour with biggest
MOO
    //cnt = contours.get(54);
    Moments = cv.moments(cnt,false);
    M00 = Moments.m00;
    M10 = Moments.m10;
    M01 = Moments.m01;
    x_cm = M10/M00;    // 75 for circle_9.jpg
    y_cm = M01/M00;    // 41 for circle_9.jpg
    XCMoutput.innerHTML = Math.round(x_cm);
    YCMoutput.innerHTML = Math.round(y_cm);
    console.log("M00 = "+M00);
    console.log("XCM = "+Math.round(x_cm));
    console.log("YCM = "+Math.round(y_cm));

    //fetch(document.location.origin+'/?xcm='+Math.round(x_cm)+';stop');
    //EXPT TEMPORARY OMIT THIS NEXT FETCH. NO EFFECT. PUT BACK
    //fetch(document.location.origin+'/?cm='+Math.round(x_cm)+';'+Math.round(
y_cm)+';stop');
    //
original//fetch(document.location.origin+'/?cm='+Math.round(x_cm)+';'+Math.ro
und(y_cm)+';'+b_tracker_int+';stop');
    //dataTransmit();
    //dataTransmit2(5);
    //GetSwitchAnalogData();
    console.log("ADCARR = " + ADCarr);
    console.log("M00ARRAY = " + M00Array);
    //ANN:14

    //**************min area bounding rect*********************
    //let rotatedRect=cv.minAreaRect(cnt);
    //let vertices = cv.RotatedRect.points(rotatedRect);
    //for(let j=0;j<4;j++){
    //    cv.line(src,vertices[j],
    //        vertices[(j+1)%4],[0,0,255,255],2,cv.LINE_AA,0);
    //}
    //***************end             min             area             bounding
rect***********************************
    //***************bounding rect*****************************
    let rect = cv.boundingRect(cnt);
    let point1 = new cv.Point(rect.x,rect.y);
    let point2 = new cv.Point(rect.x+rect.width,rect.y+rect.height);
    cv.rectangle(src,point1,point2,[0,0,255,255],2,cv.LINE_AA,0);
    //*************end bounding rect*************************
    //*************draw center point*********************
    let point3 = new cv.Point(x_cm,y_cm);
    cv.circle(src,point3,2,[0,0,255,255],2,cv.LINE_AA,0);
    //***********end draw center point*********************
    }//end if(ArgMaxArea >= 0)
    else{
      if(ArgMaxArea==-1){
        console.log("ZERO ARRAY LENGTH");
      }
      else{              //ArgMaxArea=-2
        console.log("DUPLICATE MAX ARRAY-ELEMENT");
      }
    }
```

```
    cnt.delete();
/*****************end      contours        note      cnt      line      one
up*****************************************/
   drawXCM_YCM_Text();
  }//end try
  catch{
    console.log("ERROR TRACKER NO CONTOUR");
    clear_canvas();
    clear_canvas0();
    drawErrorTracking_Text();
  }

  }//end b_tracking if statement
  else{
      XCMoutput.innerHTML = 0;
      YCMoutput.innerHTML = 0;
      fetch(document.location.origin+'/?cm='+Math.round(0)+';'+Math.round(0)+
';'+b_tracker_int+';stop');
  }
  if(mirrorimage.value==0){
   if(b_invert==false){
      cv.imshow('imageMask', mask);
   }
   else{
      cv.imshow('imageMask', mask2);
   }
   //cv.imshow('imageMask', R);
   cv.imshow('imageCanvas', src);
  }
  //ANN:8A
  src.delete();
  high.delete();
  low.delete();
  orig.delete();
  mask1.delete();
  mask2.delete();
  mask.delete();
  contours.delete();
  hierarchy.delete();
  //cnt.delete();
  RP.delete();


 /*******************END COLOR DETECT*****************************/

/***************end opencv*******************************/

 setTimeout(function(){colorDetect.click();},500);

}//end detectimage
function MaxAreaArg(arr){
    if (arr.length == 0) {
        return -1;
    }
    var max = arr[0];
    var maxIndex = 0;
    var dupIndexCount = 0; //duplicate max elements?
    if(arr[0] >= .90*aarea){
        max = 0;
```

```
    }
    for (var i = 1; i < arr.length; i++) {
        if (arr[i] > max && arr[i] < .99*aarea) {
            maxIndex = i;
            max = arr[i];
            dupIndexCount = 0;
        }
        else if(arr[i]==max && arr[i]!=0){
            dupIndexCount++;
        }
    }
    if(dupIndexCount==0){
        return maxIndex;
    }
    else{
        return -2;
    }
}//end MaxAreaArg
function clear_canvas(){
  if (mirrorimage.value==0){
    ctx.clearRect(0,0,txtcanvas.width,txtcanvas.height);
    ctx.rect(0,0,txtcanvas.width,txtcanvas.height);
    ctx.fillStyle="green";
    ctx.fill();
  }
  else{
    ctx.clearRect(0,0,txtcanvas.width,txtcanvas.height);
    ctx.rect(0,0,txtcanvas.width,txtcanvas.height);
    //ctx.fillStyle="white";
    //ctx.fill();
  }
}
function clear_canvas0(){
  if (mirrorimage.value==1){
    ctx0.clearRect(0,0,txtcanvas0.width,txtcanvas.height);
    ctx0.rect(0,0,txtcanvas0.width,txtcanvas0.height);
    ctx0.fillStyle="red";
    ctx0.fill();
  }
  else{
    ctx0.clearRect(0,0,txtcanvas0.width,txtcanvas0.height);
    ctx0.rect(0,0,txtcanvas0.width,txtcanvas0.height);
    //ctx.fillStyle="white";
    //ctx.fill();
  }
}
function drawASensorText(x,y,w,z){
  if(mirrorimage.value==0){
    ctx.fillStyle = 'black';
    ctx.font = '20px serif';
    ctx.fillText('ANA_SENSOR1_OCV = '+x,0,4*txtcanvas.height/10);
    ctx.fillText('DIG_SENSOR1_OCV                                                    =
'+y,txtcanvas.width/2,4*txtcanvas.height/10);
    ctx.fillText('ANA_SENSOR2_OCV = '+w,0,6*txtcanvas.height/10);
    ctx.fillText('DIG_SENSOR2_OCV                                                    =
'+z,txtcanvas.width/2,6*txtcanvas.height/10);
  }
  if(mirrorimage.value==1){
    ctx0.fillStyle = 'black';
```

```
    ctx0.font = '20px serif';
    ctx0.fillText('ANA_SENSOR1 = '+x,0,2*txtcanvas0.height/10);
    ctx0.fillText('DIG_SENSOR1                                                    =
'+y,txtcanvas0.width/2,2*txtcanvas0.height/10);
    ctx0.fillText('ANA_SENSOR2 = '+w,0,6*txtcanvas0.height/10);
    ctx0.fillText('DIG_SENSOR2                                                    =
'+z,txtcanvas0.width/2,6*txtcanvas0.height/10);
  }
}
function drawReadyText(){
    ctx.fillStyle = 'black';
    ctx.font = '20px serif';
    ctx.fillText('OpenCV.JS READY',txtcanvas.width/4,txtcanvas.height/10);
}
function drawColRowText(x,y){
  if (mirrorimage.value==0){
    ctx.fillStyle = 'black';
    ctx.font = '20px serif';
    ctx.fillText('ImageCols='+x,0,txtcanvas.height/10);
    ctx.fillText('ImageRows='+y,txtcanvas.width/2,txtcanvas.height/10);
  }
}
function drawRGB_PROBE_Text(){
  if (mirrorimage.value==0){
    ctx.fillStyle = 'black';
    ctx.font = '20px serif';
    ctx.fillText('Rp='+R,0,2*txtcanvas.height/10);
    ctx.fillText('Gp='+G,txtcanvas.width/4,2*txtcanvas.height/10);
    ctx.fillText('Bp='+B,txtcanvas.width/2,2*txtcanvas.height/10);
    ctx.fillText('Ap='+A,3*txtcanvas.width/4,2*txtcanvas.height/10);
  }
}
function drawXCM_YCM_Text(){
    ctx.fillStyle = 'black';
    ctx.font = '20px serif';
    ctx.fillText('XCM='+Math.round(x_cm),0,3*txtcanvas.height/10);
    ctx.fillText('YCM='+Math.round(y_cm),txtcanvas.width/4,3*txtcanvas.height
/10);
}
function drawErrorTracking_Text(){
    ctx.fillStyle = 'black';
    ctx.font = '20px serif';
    ctx.fillText('ERROR TRACKING-NO CONTOUR',0,3*txtcanvas.height/10);
}
//COPY FROM TTGO_T_JOURNAL_ROBOT_WEB_21
function dataTransmit() {
     alert("DATA_TRANSMIT");
     var xhr = new XMLHttpRequest();
     xhr.onreadystatechange = function() {
     console.log("READYSTATE===="+this.readyState);
     console.log("STATUS===="+this.status);
     if(this.readyState==4 && this.status == 200){
          //ADC = xhr.responseText;
         //ADC = xhr.response;
         console.log("RESPONSE == "+xhr.response);
         ADCarr = xhr.response;
      }
     };
```

```
     //xhr.open("GET", "/action?go=" + x, true);
     xhr.open("GET",document.location.origin+'/?cm='+Math.round(x_cm)+';'+Mat
h.round(y_cm)+';'+b_tracker_int+';stop',true);
     xhr.send();
}
//ANN:15
//NEW DATATRANSMIT
  function GetSwitchAnalogData() {
    nocache = "&nocache=" + Math.random() * 1000000;
    var request = new XMLHttpRequest();
    //document.getElementById("sw_an_data").innerHTML = "ON"; //TEST
    request.onreadystatechange = function() {
      console.log("READYSTATE===="+this.readyState);
      console.log("STATUS===="+this.status);
      if (this.readyState == 4) {
        if(this.status == 200) {
          if(this.responseText != null) {
            document.getElementById("sw_an_data").innerHTML                    =
this.responseText;
            console.log("RESPONSETEXT = "+this.responseText);
            //json_obj = JSON.parse(this.responseText);
            //console.log("DIGDATA = "+json_obj.SWITCH_STATE);
            //console.log("ANADATA = "+json_obj.ANALOG);
            json_obj = JSON.parse(this.responseText);
            dig_sensor1 = json_obj[0].DIGSENSOR1;
            dig_sensor2 = json_obj[0].DIGSENSOR2;
            ana_sensor1 = json_obj[1].ANASENSOR1;
            ana_sensor2 = json_obj[1].ANASENSOR2;
            console.log("DIGDATA1 = "+json_obj[0].DIGSENSOR1);
            console.log("DIGDATA2 = "+json_obj[0].DIGSENSOR2);
            console.log("ANADATA1 = "+json_obj[1].ANASENSOR1);
            console.log("ANADATA2 = "+json_obj[1].ANASENSOR2);
          }//end responseText
        }//end status
      }//end readyState
    }// end inner func
    //request.open("GET", "ajax_switch" + nocache, true);
    //request.open("GET", "ajax_switch", true);
    request.open("GET",document.location.origin+'/?cm='+Math.round(x_cm)+';'+
Math.round(y_cm)+';'+b_tracker_int+';stop',true);
    request.send(null);
  }//end func
//END NEW DATATRANSMIT

  </script>
  </body>
  </html>
)rawliteral";
```

BROWSER SYSTEM: XBOX CONTROLLER DESCRIPTION

The Xbox Controller program for the Browser System, depends on 2 critical imports shown at APP:1; *pygame.locals* and *dashboard_26_module*, as well as several generic imports.

#ANN:2 shows the web socket address of the ESP32 server with which the Controller will communicate two-way.

#ANN:3 are module constants and function lists used to call the GUI components.

#ANN:4 is `list0` showing the numeric state of each element (button, trigger, etc) of the controller. The ID number of each element of the list corresponds to the diagram—except for the PULSE whose purpose will be discussed below.

#ANN:5 shows the function which updates the Dashboard on the GUI using `list0` as described immediately above.

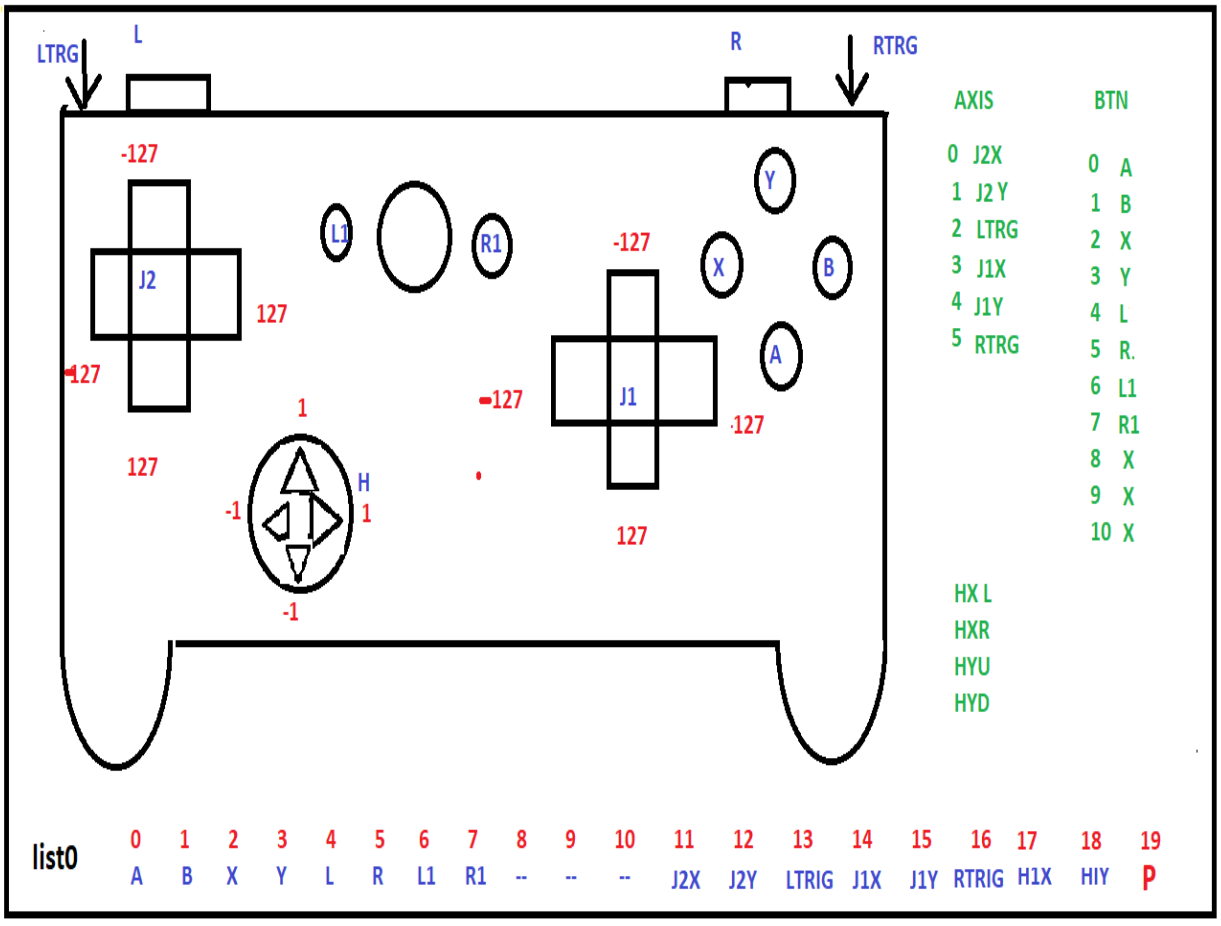
#ANN:6 Is the principal program called from main.

#ANN:7 shows the FILL which resets the colors of particular elements to prepare for the next redraw of the Controller elements. Immediately follow the FILL instructions is `list0[19]` which serves as the “pulse” of the Controller which displays a number continuously increasing by 1, transmitted to the ESP32 and then back to the Controller and displayed.

This display assures the operator that the communication link is operational.

#ANN:8 shows the collection of all Controller inputs, latching all that are designated as latched inputs, and inserts them into `list0`. (remember **ANN:5** above)

The figure below depicts `list0` contents related to the XBox inputs.



#ANN:9 shows transmission of `list0` to the ESP32 server and **#ANN:10** reception of sensor data from the server. (The sensor data is entered into a list array referenced in the dashboard update described in **ANN:5**.) To completely understand the sensor data operation of

ACKDATA1, the reader is referred to “The Zero Hack” section of this paper.

BROWSER STREAMING SYSTEM: XBOX CONTROLLER CODE

IMPORTANT NOTE: The *dashboard_26_module.py* import in the following code is given in the XBOX CONTROLLER PROGRAM chapter.

ROBOT_BROWSER_XBOX_P.py

You can download the code on the following link:

- https://github.com/arshacker/ESP32-ROBOT-CAM-BROWSER-XBOX-CONTROLLER-OPENCV-SYSTEM/blob/main/ROBOT_BROWSER_XBOX_P.py

```
#ANN:1
import pygame
from pygame.locals import *
import dashboard_26_module
import time
import datetime
import math
import keyboard
#ANN:2
import websocket  #actually websocket-client; websocket gives wrong WebSocket()

ws = websocket.WebSocket()
ws.connect("ws://X.X.X.X:82/ws") #ROBOT_BROWSER_SERVER IP ADDRESS

i = 0

listBIN = [0,0,0,0,0,0,0]

def DecToBin(num,i):

    if num > 1:
        DecToBin(num // 2,i+1) #push down stack, first in-last out
    #print("num = ",num)
    #print("num%2=",num % 2)
    #print("i=",i)
    listBIN[6-i]=num % 2  #6-i reverses so last out goes into listBIN[0]
```

```
    #print(listBIN)



index = 0

BYTE_NUMB = 127
#ANN:3
JStick_Arc_Radius = dashboard_26_module.radius_1

listFunc = [dashboard_26_module.Button_A,dashboard_26_module.Button_B,\
            dashboard_26_module.Button_X,dashboard_26_module.Button_Y,\
            dashboard_26_module.Button_L,dashboard_26_module.Button_R,\
            dashboard_26_module.Button_L1,dashboard_26_module.Button_R1,\
            0,0,0,\
            dashboard_26_module.JStick_2X,dashboard_26_module.JStick_2Y,\
            dashboard_26_module.TrigL_Sig,\
            dashboard_26_module.JStick_1X,dashboard_26_module.JStick_1Y,\
            dashboard_26_module.TrigR_Sig,\
            dashboard_26_module.HatX,dashboard_26_module.HatY]

listSensorFunc = [dashboard_26_module.SENSOR_1_VALUE,\
                  dashboard_26_module.SENSOR_2_VALUE,\
                  dashboard_26_module.SENSOR_3_VALUE]
#ANN:4
list0 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
#list0 = [0,1,1,0,1,0,1,1,0,0,0,117,-122,1,127,-127,1,-1,1,0]
#        0 1 2 3 4 5 6 7 X X X 11  12  13   14  15  16    17 18  19
#                              J2X J2Y LTRG J1X J1Y RTRG  HX HY  PULSE
#                               + -
#                              0-127

                #ascii [=91  ]=93  ,=44  sp=32  -=45  0=48

listACK1 = [0,0,0,0,0]
listACK2 = [0,0,0,0]

list_rcv = [0,0,0,0,0,0,0,0,0,0]

listACK_rcv = [0,0,0,0,0,0,0,0,0,0]

listACK_rcv_ord = [0,0,0,0,0,0,0,0,0,0]  #only first 4 locs used

button_out = [0,0,0,0,0,0,0,0,0,0,0]
button_out_prev = [0,0,0,0,0,0,0,0,0,0,0]
button_prev_zero = 0
button_out_diff = [0,0,0,0,0,0,0,0,0,0,0]

running = True

black = (0,0,0)
gray = (128,128,128)
light_gray = (200,200,200)
dark_gray = (64,64,64)
```

```
white = (255,255,255)
red = (255,0,0)
light_red =(128,0,0)
green = (0,255,0)
blue = (0,0,255)
listColor_in=[0,0,0]
listColor_in[0]=abs(-255)
Color_in = tuple(listColor_in)

Max_Arc_Width = math.floor((dashboard_26_module.ScreenWidth/250)*25) #25
Max_JStick_Amplitude = 127
#ANN:5
def update_dashboard():
    for i in [0,1,2,3,4,5,6,7,11,12,13,14,15,16,17,18]:
        if(i<8):
            if(list0[i]==1):
               listFunc[i](green)
            if(list0[i]==0):
               listFunc[i](gray)
        if(i==13 or i == 16):
            #MAPPING FOR NEG X   RED=127+ABS(X)  GREEN=127-ABS(X)
            #        FOR POS X   RED=127-ABS(X)  GREEN=127+ABS(X)
            if(list0[i]<0):
               listFunc[i]((127+abs(list0[i]),127-abs(list0[i]),0))
            if(list0[i]>=0):
               listFunc[i]((127-abs(list0[i]),127+abs(list0[i]),0))
        if(i==11 or i==12 or i==14 or i==15):
            if(list0[i]<0):
               #ARC WIDTH = FRACTIONAL (MAX AMPL-AMPL)/MAX AMPL
               listFunc[i](math.floor(((Max_JStick_Amplitude-abs(list0[i]))/\
                 Max_JStick_Amplitude)*Max_Arc_Width),Max_Arc_Width)
            if(list0[i]>=0):
               #ARC WIDTH = FRACTIONAL (MAX AMPL-AMPL)/MAX AMPL
               listFunc[i](Max_Arc_Width,\
                 math.floor(((Max_JStick_Amplitude-abs(list0[i]))/\
                 Max_JStick_Amplitude)*Max_Arc_Width))
        if(i==17 or i==18):
            if(list0[17]==1 and list0[18]==1):
               listFunc[17](gray,green)
               listFunc[18](green,gray)
            if(list0[17]==1 and list0[18]==0):
               listFunc[17](gray,green)
               listFunc[18](gray,gray)
            if(list0[17]==1 and list0[18]==-1):
               listFunc[17](gray,green)
               listFunc[18](gray,red)
            if(list0[17]==0 and list0[18]==1):
               listFunc[17](gray,gray)
               listFunc[18](green,gray)
            if(list0[17]==0 and list0[18]==0):
               listFunc[17](gray,gray)
               listFunc[18](gray,gray)
            if(list0[17]==0 and list0[18]==-1):
               listFunc[17](gray,gray)
```

```
                listFunc[18](gray,red)
            if(list0[17]==-1 and list0[18]==1):
                listFunc[17](red,gray)
                listFunc[18](green,gray)
            if(list0[17]==-1 and list0[18]==0):
                listFunc[17](red,gray)
                listFunc[18](gray,gray)
            if(list0[17]==-1 and list0[18]==-1):
                listFunc[17](red,gray)
                listFunc[18](gray,red)
    for j in [0,1,2]:
            if (j==0):
                 listSensorFunc[j](green,listACK2[j]/127)
            if (j==1):
                 listSensorFunc[j](green,listACK2[j]/127)
            if (j==2):
                 listSensorFunc[j](red,listACK2[j])
#ANN:6
def do_running():
  global i
  global running
  global index
  global listACK1
  while (running):
    start_time = datetime.datetime.now()
    #time.sleep(4)
    #ANN:7
    dashboard_26_module.JStick_1(red,green,red,green)
    dashboard_26_module.JStick_2(red,green,red,green)
    dashboard_26_module.SENSOR_1_FILL(gray,-80,260)
    dashboard_26_module.SENSOR_2_FILL(gray)
    dashboard_26_module.SENSOR_3_FILL(green)
####################FROM XBOX_TW0_TEST_8_EXPT2############
    list0[19] = index    #pulse of the transmission sytem
    index = index % BYTE_NUMB
    index = index + 1
    #print("List0 = ",list0)
    #ANN:8
  ###GET CONTROLLER INPUTS#################
   # Get count of joysticks
    joystick_count = pygame.joystick.get_count()
    #print(joystick_count)
    for i in range(joystick_count):
        joystick = pygame.joystick.Joystick(i)
        joystick.init()

        # Get the name from the OS for the controller/joystick
        name = joystick.get_name()

       # Usually axis run in pairs, up/down for one, and left/right for the other.
        axes = joystick.get_numaxes()
        #print(axes)
        for j in range( axes ):
            axis = joystick.get_axis( j )
```

```
            list0[j+11] = round(BYTE_NUMB*axis)

        buttons = joystick.get_numbuttons()

        for i in range( buttons ):
            button_out[i] = joystick.get_button( i )
            if(button_out[i]==1 and button_out_prev[i]==0):
                button_out_diff[i] =1
            else:
                button_out_diff[i] =0
            button_out_prev[i] = button_out[i]
            list0[i] = button_out_diff[i]^list0[i]


        # Hat switch. All or nothing for direction, not like joysticks.
        # Value comes back in a tuple.
        hats = joystick.get_numhats()

        for i in range( hats ):
            hat = joystick.get_hat(i )
            mytuple = hat
        list0[17] = mytuple[0]
        list0[18] = mytuple[1]


    print("list0 = ",list0)
######################END FROM XBOX_TW0_TEST_8_EXPT2########

    update_dashboard()
    pygame.display.update()
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            print("QUIT")
            running = False
            pygame.quit()

###############PREPARE AND SEND MESSAGE#######################
    #ANN:9
    string0 = str(list0)
    string0_stripped = string0[1:len(string0)-1] #strip [ and ]
    message = string0_stripped.encode()
    print("message =",message)
    #message = bytes(list1)
    #ws.send("hello") #works. sends each char as ascii
    ws.send(message)   #workd. sends each nume ral as ascii
    #ws.send(5) #does not work
    #ws.send_binary(100|35|50)  #does not work
    #end_time1 = datetime.datetime.now()
    #exec_time1 = end_time1 - start_time
    #print("execTime1 = ",exec_time1)
    index1 = 0
    data = b""
    msgLength = len(message)
    print("msgLength = ",msgLength)
```

```
############END PREPARE AND SEND MESSAGE#######################

#----------ALT RCV MSG-------------------------------
    for x in range(7):
        listBIN[x] = 0
    print("listBIN = ",listBIN)

    for x in range(5):
        listACK1[x] = 0
        #listACK2[x] = 1
    print("listACK1 = ",listACK1)

    for x in range(4):
        listACK2[x] = 1
    print("listACK2 = ",listACK2)

    #for z in range(10):
        #listACK_rcv[z] = 250
    #print("listACK_rcv = ",listACK_rcv)


    #ANN:10
    ACKDATA1 = ws.recv()
    print("ACKDATA1 = ",ACKDATA1)
    listACK1 = list(ACKDATA1)
    #print(list(ACKDATA1))
    print("listACK1 = ",listACK1)
    DecToBin(listACK1[4],i)
    print("DECTOBIN = ",listBIN)
    if listBIN[0]==1:
        listACK2[0]=0
    else:
        listACK2[0]=listACK1[0]
    if listBIN[1]==1:
        listACK2[1]=0
    else:
        listACK2[1]=listACK1[1]
    if listBIN[2]==1:
        listACK2[2]=0
    else:
        listACK2[2]=listACK1[2]
    if listBIN[3]==1:
        listACK2[3]=0
    else:
        listACK2[3]=listACK1[3]
    print("listACK2 = ",listACK2)

    #print("listACK2 =",listACK2)

    end_time2 = datetime.datetime.now()
    exec_time2 = end_time2 - start_time
    print("execTime2 = ",exec_time2)
```

```
#----------END ALT RCV MSG--------------------------

def main():
  dashboard_26_module.Dashboard()

  # Used to manage how fast the screen updates
  #clock = pygame.time.Clock()

  # Initialize the joysticks
  pygame.joystick.init()

###############COMM  INIT##############################
#import socket
#sock = socket.socket()
#host = "192.168.1.9" #ESP32 IP in local network
#port = 80             #ESP32 Server Port
#sock.connect((host, port))


  #ws = websocket.WebSocket()
  #ws.connect("ws://192.168.1.4:82/ws")

###################END COMM INIT#########################

#######################COMM ESTABLISHED####################

  ACKDATA = ws.recv()
  print("ACKDATA = ",ACKDATA)
  print("PRESS q TO SEND AND z TO CLOSE")

#if(keyboard.is_pressed('q')):

####################END COMM ESTABLISHED####################

  do_running()

if __name__=="__main__":
  main()
```

PYTHON STREAMING SYSTEM PROGRAM

SERVER PROGRAM DESCRIPTION

Some of the code used in this Server is found on GitHub: Dallyla/delorean-express: In this project ESP32 cam is used for color detecting with openCV and Python. The Arduino code related to the Camera is not mine; you can see the credits at the top of the code.

PYTHON STREAMING SERVER

The program listing for this server is given in the next section. The following is an explanation of the key points in the program. The listing is annotated and the explanation refers to these annotations. The reader can copy the code in PART 2 into an Arduino IDE and use its FIND function to refer to each annotation.

The key to this program is seen in the inclusion of the following two header files: `ESPAsyncWebServer_ARS.h` and `esp_http_server.h` as seen at **ANN 1**. The former header file is a minor but critical modification of a section of the `ESPAsyncWebServer.h` as shown below.

Before modification:

```
#ifndef WEBSERVER_H
typedef enum {
  HTTP_GET     = 0b00000001,
  HTTP_POST    = 0b00000010,
  HTTP_DELETE  = 0b00000100,
  HTTP_PUT     = 0b00001000,
  HTTP_PATCH   = 0b00010000,
  HTTP_HEAD    = 0b00100000,
  HTTP_OPTIONS = 0b01000000,
  HTTP_ANY     = 0b01111111,
#endif
```

After modification:

```
#ifndef WEBSERVER_H
typedef enum {
  //HTTP_GET     = 0b00000001,
  //HTTP_POST    = 0b00000010,
  //HTTP_DELETE  = 0b00000100,
  //HTTP_PUT     = 0b00001000,
  //HTTP_PATCH   = 0b00010000,
  //HTTP_HEAD    = 0b00100000,
  //HTTP_OPTIONS = 0b01000000,
  HTTP_ANY     = 0b01111111,
#endif
```

The modification is accomplished by simply locating the above typedef enum code block and performing the indicated modification.

By **commenting out** the redefinitions of HTTP_GET, etc., the program will compile with the two headers. The former header is the server for the XBOX Controller client while the latter header serves the python window client.

`ESPAsyncWebServer_ARS.h`, and the Espressif headers `dl_lib.h`, `dl_libcoeffgetter_if.h`, , `dl_lib_conv_queue.h`, , `dl_lib_conq_queue.h`, `dl_lib_matrix.h`, and `dl_lib_matrixq.h` can be found at

- GitHub: arshacker/ESP32-ROBOT-CAM-VIDEO-XBOX-CONTROLLER-OPENCV-SYSTEM
- ESP32-ROBOT-CAM-VIDEO-XBOX-CONTROLLER-OPENCV-SYSTEM/ROBOT_STREAMING_SERVER_P at main · arshacker/ESP32-ROBOT-CAM-VIDEO-XBOX-CONTROLLER-OPENCV-SYSTEM · GitHub

ANN 2 introduces `Wire.h` which is the I2C library, SDA and SCL data and clock pins for I2C, arrays for data communication as described earlier in the Streaming System description chapter, and global variables for the centroid of the OpenCV color target also described in that chapter.

ANN 3 shows the creation of the port and socket handler address for the XBOX Controller.

ANN 4 is the I2C code and data array structures needed to send and receive messages between the ESP32 and the Arduino Mega as described in in the Streaming System description chapter.

ANN 5 `codePrep` is described in a special chapter entitled “The Zero Hack”.

ANN 6 shows the function which formats the data return from the server to the XBOX Controller client and again `codePrep` is described in “the Zero Hack” chapter.

ANN 7 show the `onWsEvent` which fires upon receipt of a socket message from the XBOX Controller and returns a message back. The latter message contains sensor data received from the Arduino Mega via I2C as described in in the Streaming System description chapter.

ANN 8 Message parser extracts the commas from the message received from the XBOX Controller.

ANN 10 Introduces the user's SSID and password into to the network.

ANN 11 is the section defining the T-Journal ESP32 and Camera as well as the stream handler and the `startCameraServer` at default port 80 of the IP Address. The server then continuously transmits to the python window in the client when the client signs into the IP Address.

ANN 12 shows the setup which is fairly self-explanatory for readers with ESP Camera experience. The reader should note that near the end of the setup, the IP address is prepared for I2C transmission (to the PRO MINI as described in the Streaming System description chapter and finally the server handler for the XBOX Controller is activated.

ANN 13 Typical of such "on event" programs, the loop is empty...except in this case, a one-time I2C transmission of the ESP32 IP address is sent to the Pro Mini as described in the Streaming System description chapter.

This ends the description of the ESP32 SERVER for PYTHON STREAMING WINDOW & XBOX CONTROLLER-FOR PYTHON STREAMING program.

PYTHON STREAMING SERVER CODE

ROBOT_STREAMING_SERVER_P.ino

You can download the code at the following link:

- https://github.com/arshacker/ESP32-ROBOT-CAM-VIDEO-XBOX-CONTROLLER-OPENCV-SYSTEM/blob/main/ROBOT_STREAMING_SERVER_P/ROBOT_STREAMING_SERVER_P.ino

```
/*********
  Parts of this code are based on the following reference.
  Rui Santos
  https://RandomNerdTutorials.com

  IMPORTANT!!!
   - Select Board "ESP32 Wrover Module"
   - Select the Partition Scheme "Huge APP (3MB No OTA)

  Permission is hereby granted, free of charge, to any person obtaining a copy
  of this software and associated documentation files.
  The above copyright notice and this permission notice shall be included in
all
  copies or substantial portions of the Software.
  Some of the code, particularly ANN:12 below, in this program is from
  https://github.com/Dallyla/delorean-express
*********/
//ANN 1
#include "esp_camera.h"
#include <WiFi.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "soc/soc.h" //disable brownout problems
#include "soc/rtc_cntl_reg.h"  //disable brownout problems
#include "dl_lib.h"
#include "esp_http_server.h"

#include <AsyncTCP.h>
#include <ESPAsyncWebServer_ARS.h> //TOOK OUT REDEFS OF GET POST ETC IN ORIGINAL

//#define SOFTAP

/********************FROM OCV_COLORTRACK11_MS_34_3************/
//ANN 2
#include <Wire.h>    //comm to arduino
#define myWire Wire
#define SDA 14
#define SCL 13

#define SCREENCOLS 400
```

```
#define SCREENROWS 300

#define CODE_LEN 7
int code_array[CODE_LEN] = {0,0,0,0,0,0,1};
int code_nbr = 1;

//#define MSG_LEN 7  //
#define MSG_INT_LEN 5
#define MSG_LEN  2*MSG_INT_LEN-1

//char msg[MSG_LEN] = {0,};
//int msg_int[4] = {97,98,127,1}; // {97,98,127};

//char msg[MSG_LEN] = {0,};
char msg[MSG_INT_LEN] = {0,};
int msg_int[MSG_INT_LEN] = {97,98,127,1}; // {97,98,127};
                    //ALLOW POS INTEGERS 0-127

int One_Time_Transmit = 1;

int lock = 0;
char cTT[256] = {0,};
//char c[] = {0,};
byte d[256] = {0,};
char e;

int lenGlobal=0;

byte bufIP[4] = {0,0,0,0};  //i2c xmit
byte bufToBot[20] = {0,};  //bufToBot[19] = {0,};
byte buffer[40];             //i2c rcv(prev said transmit?
byte buffer_temp[40];    //check contents before xfr to buffer
int n = 0;                   //i2c rcv

int finalIndex = 0;
int initialIndex = 0;
int kIndex = 0;

long timeStart;
long timeFinish;
long timeStart1;
long timeFinish1;

int DELTA_XCM = 0;
int DELTA_YCM = 0;
byte b_Tracker = 0;
byte b_DELTA_XCM = 0;
byte b_DELTA_YCM = 0;

//ANN 3
AsyncWebServer server3(82);

AsyncWebSocket ws("/ws");

/************************start                                    i2c
transmit*********************************/
//ANN 4
  void i2cTransmit(void){
               /************SEND I2C DATA*********************/
```

```
    //Serial.println(F("Test with 1 transmissions of writing 10 bytes each"));
  //byte buf[20] = { 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, };
  byte  buf[27]  =  {  100,  101,  102,  103,  104,  105,  106,  107,  108,
109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125,126};
  int err = 0;
  unsigned long millis1 = millis();
  boolean firsterr = false;
  //for( int i=0; i<1; i++)
  //{
    Serial.println(F("Sending data"));
    timeStart1 = millis();
    myWire.beginTransmission(4);

    for(int z=0; z<=3;z++){
      buf[z]=bufIP[z];
    }
    for(int v=0; v<=19; v++){
      buf[v+4]=bufToBot[v];
    }
    buf[24] = b_DELTA_XCM;
    buf[25] = b_DELTA_YCM;
    buf[26] = b_Tracker;

    Serial.print("BUF = ");
    for( int k=0; k<27; k++)
    {
    Serial.print( (int) buf[k]);
    Serial.print(F(", "));
    }
    Serial.println();

    myWire.write( buf, 27);       //replace with bufToBot from client
    //myWire.write(bufToBot,5);
    if( myWire.endTransmission() != 0)
    {
      err++;
    }
    delayMicroseconds(100);  // Even the normal Arduino Wire library needs some
delay when the Slave disables interrupts.
 // }

     myWire.beginTransmission(6);
     myWire.write( buf, 27);
    if( myWire.endTransmission() != 0)
    {
      err++;
    }
    delayMicroseconds(100);  // Even the normal Arduino Wire library needs some
delay when the Slave disables interrupts.


  unsigned long millis2 = millis();
  Serial.print(F("total time: "));
  Serial.print(millis2 - millis1);
  Serial.print(F(" ms, total errors: "));
  Serial.println(err);

  delay(2);
```

```
  timeFinish1 = millis();
  Serial.print("INNER TIME  = ");
  Serial.println(timeFinish1-timeStart1);

                              /**END SEND I2C DATA*****/

                              /**REQUEST I2C DATA******/

  Serial.println(F("Requesting data"));
  n = myWire.requestFrom(4, 10);    // request bytes from Slave //make n a
global
  Serial.print(F("n="));
  Serial.print(n);
  Serial.print(F(", available="));
  Serial.println(myWire.available());

//   myWire.printStatus(Serial);       //  This  shows  information  about  the
SoftwareWire object.

//  byte buffer[40];          //make this a global for access by cmd func
//  for( int j=0; j<n; j++)
//    buffer[j] = myWire.read();
  myWire.readBytes( buffer_temp, n);

  Serial.print("RCV_BUFFER_TEMP = ");
  for( int k=0; k<n; k++)
  {
    if( k == 0)
      Serial.print(F("*"));      // indicate the number of the counter
    Serial.print( (int) buffer_temp[k]);
    Serial.print(F(", "));
  }
  Serial.println();

  if(buffer_temp[0]<128){                //NO ERROR
    for( int k=0; k<n; k++){
      buffer[k] = buffer_temp[k];
    }
  }

  Serial.print("RCV_BUFFER = ");
  for( int k=0; k<n; k++)
  {
    if( k == 0)
      Serial.print(F("*"));      // indicate the number of the counter
    Serial.print( (int) buffer[k]);
    Serial.print(F(", "));
  }
  Serial.println();

                                /***END              REQUEST              I2C
DATA*********/
  delay(2);

  }

 /**********************end i2c transmit*********************************/
//ANN 5
void codePrep(void){
```

```
  code_nbr = 2*2*2*2*2*2*code_array[0] + 2*2*2*2*2*code_array[1] +
             2*2*2*2*code_array[2] + 2*2*2*code_array[3]+
             2*2*code_array[4] + 2*code_array[5] + code_array[6];
  Serial.print("CODE NBR = ");
  Serial.println(code_nbr);
}

//ANN 6
void msgPrep(void){
  //takes msg_int[3]={X,Y,Z] and creates msg[X,',',Y,',',Z} WHERE MSG_LEN = 5
/*
  for(int i = 1; i < MSG_LEN-2; i = i + 1){
    msg[(2*i)-1] = ',';  //comma separators for python
                         //msg[1],msg[3]...
  }
  for(int j = 1; j < MSG_LEN-1; j = j + 1){
    if(buffer[(j-1)]!=0){
        msg[(2*j)-2] = buffer[(j-1)];
    }
    else{
        msg[(2*j)-2] = 1;     //cant transmit 0 to python?????
    }
    //msg[(2*j)-2] = msg_int[(j-1)]; //prepared message
      //msg[0]=msg_int[0],msg[2]=msg_int[1],msg[4]=msg_int[2]...
  }
 */
 //below is most recent before new strategy
 /*
  for(int i = 1; i<MSG_INT_LEN+1; i = i + 1){
    msg[(2*i)-1] = 2;  //MARKER 2 SAYS IT IS NOT ZERO
  }
  for(int i = 0; i<MSG_INT_LEN; i = i + 1){
    if (buffer[i]!=0){
      msg[2*i]=buffer[i];
    }
    else{
      msg[2*i]=1;  //cant transmit 0 (null) to python
      msg[(2*i)+1]=3;  //MARKER 3 SAYS IT SHOULD BE ZERO
    }
  }
  */

  for(int j=0; j<CODE_LEN-1; j=j+1){   //leave last location as 1
    code_array[j] = 0;                 //code_array(CODE_LEN-1)==1
  }

  for(int i = 0; i<MSG_INT_LEN; i = i + 1){
    if (buffer[i]!=0){
      msg[i]= buffer[i];
      code_array[i] = 0;
    }
    else{
      msg[i] = 1;  //cant transmit 0 (null) to python
      code_array[i] = 1;
    }
  }

  codePrep();
  msg[MSG_INT_LEN-1] = (char)code_nbr; //put code into MSG_INT_LEN location
```

```
  //Serial.print("BUFFER BUFFERR BUFFER ===");
  //Serial.println(buffer[1]); //prints a
  //msg[2] = '\x01';
  //msg[1] = 5;
  //msg[0] = buffer[0];
  //msg[2] = buffer[1];
  //msg[4] = buffer[2];
  //msg[6] = buffer[3];
}

//ANN 7
void  onWsEvent(AsyncWebSocket  *  server3,  AsyncWebSocketClient  *  client,
AwsEventType type, void * arg, uint8_t *data, size_t len){

  //msgPrep(); //comma separators  //put in ws_evt_data

  //int msg_int[] = {97,',',98,',',127}; //commas separators useful in python
  //char msg[5] = {0,};
  //char msg[] = {'a','b','c'};
  //char msg[] = {97,98,99};
  //char msg[] = {97,',',98,',',99};
  //char msg[] = {97,',',98,',',31};

  //msg[0] = msg_int[0]; //int to char for transmission
  //msg[1] = msg_int[1];
  //msg[2] = msg_int[2];
  //msg[3] = msg_int[3];
  //msg[4] = msg_int[4];

  lenGlobal = len;

  if(type == WS_EVT_CONNECT){

    Serial.println("Websocket client connection received");
    client->text("Hello from ESP32 Server3");
    //client->binary(msg); //move to ws_evt_data

  } else if(type == WS_EVT_DISCONNECT){
    Serial.println("Client disconnected");

  }else if(type==WS_EVT_DATA){
    Serial.println("Data received: ");

    for(int j=0; j < len; j++){
      d[j] = data[j];          //data read out only once allowed
      cTT[j] = d[j];  //ascii to char


      Serial.print(cTT[j]);   //TIMING
    }//end for j loop
    Serial.println();
    Serial.println(len);



    for(int i=0; i < len; i++){
      Serial.print(d[i]);     //TIMING
      Serial.print("|");
```

```
    }
    Serial.println();

    parse_msg();           //convert msg from python to array format
    initializeTT();


    msgPrep(); //comma separators
    client->binary(msg);  //return msg to python

    Serial.println("I2C Transmit");
    i2cTransmit();
    initializeTT();

   }//end else if ws evt data
}//end onwsevent


void initializeTT(void)
{
  for(int n=0;n<256;n++){
    cTT[n] = 0;
    d[n] = 0;
   }
}

/*
void initialize(void)
{
  for(int n=0;n<255;n++)
  {c[n] = 0;}
}
*/

//ANN 8
void parse_msg(void){

   Serial.print("cTT = ");
   for(int w=0; w<lenGlobal;w=w+1){
      Serial.print(cTT[w]);
   }
   Serial.println();

   String var = String(cTT);
   var = String('[') + var + String(']');
   Serial.print("var = ");
   Serial.println(var);

    /*******************parse on comma*********************************/
  /*******************orig parse*************************
  int index1 = var.indexOf(',',0);
  int index2 = var.indexOf(',',index1+1);
  int index3 = var.indexOf(',',index2+1);
  int index4 = var.indexOf(',',index3+1);
  int index5 = var.indexOf(',',index4+1);

  int indexEND = var.indexOf('?');  //will return -1
  Serial.println(indexEND);
  String X_VALUE = var.substring(1,index1); //omit the [
```

```
  String Y_VALUE = var.substring(index1+1,index2);
  String A_VALUE = var.substring(index2+1,index3);
  String B_VALUE = var.substring(index3+1,index4);
  String C_VALUE = var.substring(index4+1,index5);
  Serial.println(X_VALUE);
  Serial.println(Y_VALUE);
  Serial.println(C_VALUE);
  int x_value_int = X_VALUE.toInt();
  x_value_int++;
  Serial.println(x_value_int);
  bufToBot[0] = X_VALUE.toInt();
  bufToBot[1] = Y_VALUE.toInt();
  bufToBot[2] = A_VALUE.toInt();
  bufToBot[3] = B_VALUE.toInt();
  //bufToBot[4] = C_VALUE.toInt(); //SEND COUNT TO BOT only goes to 255,then
starts again
  **********end orig parse**********/

  /**********new parse**********/
  initialIndex = 1;  //IGNORE THIS start initialIndex at 1 b/c python transmits
[ .
  kIndex = 0;
  finalIndex = 0;
  while(finalIndex!=-1){
    finalIndex = var.indexOf(',',initialIndex);
    bufToBot[kIndex] = var.substring(initialIndex,finalIndex).toInt();
    //Serial.println(bufToBot[kIndex]);
    if(finalIndex==-1){
      //Serial.println(bufToBot[kIndex]);
      initialIndex = 0;
      finalIndex = 0;
      kIndex = 0;
      break;}
    initialIndex = finalIndex+1;
    kIndex++;
  }
/**********end new parse**********/
  Serial.print("BUFTOBOT = ");
  for(int m=0;m<20;m++){      //NEW
    Serial.print(bufToBot[m]);
    Serial.print(" ");
  }
  Serial.println();
    /**********end parse on comma**********/
}

/**********END FROM OCV_COLORTRACK11_MS_34_3**********/

//Replace with your network credentials

//ANN 10
const char* ssid = "ssid";
const                    char*                    password                    =
"password";

const char* ssidAP = "ssidAP"; // ars  soft AP
const char* passwordAP = "passwordAP"; //ars softAP
```

```
#define PART_BOUNDARY "123456789000000000000987654321"

// This project was tested with the AI Thinker Model, M5STACK PSRAM Model and
M5STACK WITHOUT PSRAM
#define CAMERA_MODEL_T_JOURNAL
//#define CAMERA_MODEL_AI_THINKER
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WITHOUT_PSRAM

// Not tested with this model
//#define CAMERA_MODEL_WROVER_KIT

/*#if defined(CAMERA_MODEL_WROVER_KIT)
  #define PWDN_GPIO_NUM    -1
  #define RESET_GPIO_NUM   -1
  #define XCLK_GPIO_NUM    21
  #define SIOD_GPIO_NUM    26
  #define SIOC_GPIO_NUM    27

  #define Y9_GPIO_NUM      35
  #define Y8_GPIO_NUM      34
  #define Y7_GPIO_NUM      39
  #define Y6_GPIO_NUM      36
  #define Y5_GPIO_NUM      19
  #define Y4_GPIO_NUM      18
  #define Y3_GPIO_NUM       5
  #define Y2_GPIO_NUM       4
  #define VSYNC_GPIO_NUM   25
  #define HREF_GPIO_NUM    23
  #define PCLK_GPIO_NUM    22
#elif defined(CAMERA_MODEL_M5STACK_PSRAM)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     25
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       32
  #define VSYNC_GPIO_NUM    22
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21
#elif defined(CAMERA_MODEL_M5STACK_WITHOUT_PSRAM)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     25
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
```

```
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       17
  #define VSYNC_GPIO_NUM    22
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21
#elif defined(CAMERA_MODEL_AI_THINKER)
  #define PWDN_GPIO_NUM     32
  #define RESET_GPIO_NUM    -1
  #define XCLK_GPIO_NUM      0
  #define SIOD_GPIO_NUM     26
  #define SIOC_GPIO_NUM     27

  #define Y9_GPIO_NUM       35
  #define Y8_GPIO_NUM       34
  #define Y7_GPIO_NUM       39
  #define Y6_GPIO_NUM       36
  #define Y5_GPIO_NUM       21
  #define Y4_GPIO_NUM       19
  #define Y3_GPIO_NUM       18
  #define Y2_GPIO_NUM        5
  #define VSYNC_GPIO_NUM    25
  #define HREF_GPIO_NUM     23
  #define PCLK_GPIO_NUM     22
#elif defined(CAMERA_MODEL_T_JOURNAL)*/

//ANN 11
//       T-JOURNAL                    AI_THINKER
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM     27     //         0
#define SIOD_GPIO_NUM     25     //        26
#define SIOC_GPIO_NUM     23     //        27

#define Y9_GPIO_NUM       19     //        35
#define Y8_GPIO_NUM       36     //        34
#define Y7_GPIO_NUM       18     //        39
#define Y6_GPIO_NUM       39     //        36
#define Y5_GPIO_NUM        5     //        21
#define Y4_GPIO_NUM       34     //        19
#define Y3_GPIO_NUM       35     //        18
#define Y2_GPIO_NUM       17     //         5
#define VSYNC_GPIO_NUM    22     //        25
#define HREF_GPIO_NUM     26     //        23
#define PCLK_GPIO_NUM     21     //        22
/*#else
  #error "Camera model not selected"
#endif*/

//ANN 12
static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary="
PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length:
%u\r\n\r\n";

httpd_handle_t stream_httpd = NULL;
```

```
static esp_err_t stream_handler(httpd_req_t *req){
  camera_fb_t * fb = NULL;
  esp_err_t res = ESP_OK;
  size_t _jpg_buf_len = 0;
  uint8_t * _jpg_buf = NULL;
  char * part_buf[64];

  res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
  if(res != ESP_OK){
    return res;
  }

  while(true){
    fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Camera capture failed");
      res = ESP_FAIL;
    } else {
      if(fb->width > 400){
        if(fb->format != PIXFORMAT_JPEG){
          bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
          esp_camera_fb_return(fb);
          fb = NULL;
          if(!jpeg_converted){
            Serial.println("JPEG compression failed");
            res = ESP_FAIL;
          }
        } else {
          _jpg_buf_len = fb->len;
          _jpg_buf = fb->buf;
        }
      }
    }
    if(res == ESP_OK){
      size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
      res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
    }
    if(res == ESP_OK){
      res         =         httpd_resp_send_chunk(req,         _STREAM_BOUNDARY,
strlen(_STREAM_BOUNDARY));
    }
    if(fb){
      esp_camera_fb_return(fb);
      fb = NULL;
      _jpg_buf = NULL;
    } else if(_jpg_buf){
      free(_jpg_buf);
      _jpg_buf = NULL;
    }
    if(res != ESP_OK){
      break;
    }
    //Serial.printf("MJPG: %uB\n",(uint32_t)(_jpg_buf_len));
  }
  return res;
}
```

```
void startCameraServer(){
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  config.server_port = 80;

  httpd_uri_t index_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
  };

  //Serial.printf("Starting web server on port: '%d'\n", config.server_port);
  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &index_uri);
  }
}

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector

  Serial.begin(115200);
  Serial.setDebugOutput(false);
  Serial.println();

  myWire.begin(SDA,SCL);

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;

  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }
```

```
  // Camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
 // Wi-Fi connection

 #if defined(SOFTAP)
/******************SOFTAP*********************************/
  WiFi.softAP(ssidAP,NULL,1,0,1);
  // ssid,pwd,channel(1-13), broadcast/hidden, max connections(4)
  Serial.print("Setting AP..");
  IPAddress ip = WiFi.softAPIP();

  /*
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  */
  Serial.print("softIP = ");
  Serial.println(ip);

  Serial.println("");
  Serial.println("WiFi connected");

  Serial.print("Camera Stream Ready! Go to: http://");
  //Serial.println(WiFi.localIP()); ars using softAP
  //Serial.println(IP);
/***********************END SOFTAP*******************************/


 #else
 /***************STATION POINT***********************/
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  Serial.print("Camera Stream Ready! Go to: http://");
  Serial.print(WiFi.localIP());
  IPAddress ip = WiFi.localIP();
   /********************END STATION POINT************************/

 #endif

  /********************FROM OCV_COLORTRACK11_MS_34_3************/
    /********************transmit ip address*****************************/


    //bufIP[0] = ip[0];
  //Serial.println(String(ip[0]));
  String ipString = String(ip[0])+String('.')+String(ip[1])+String('.')+
                    String(ip[2])+String('.')+String(ip[3]);
  Serial.println(ipString);
```

```
  int z = String(ip[0]).toInt();
  z=z+1;
  Serial.println(z);
  bufIP[0] = (byte)String(ip[0]).toInt();  //bufIP[] is byte array
  Serial.println(bufIP[0]);
  for(int k=0; k<=3; k++){
    //bufIP[k] = (byte)String(ip[k]).toInt();  //bufIP[] is byte array
    bufIP[k] = String(ip[k]).toInt();  //bufIP[] is byte array
  }
  //Serial.println((int)bufIP[2]);
  Serial.println(bufIP[1]);
  Serial.println(bufIP[2]);
  Serial.println(bufIP[3]);

  /**********************end                    transmit                    ip
address***********************************/


  ws.onEvent(onWsEvent);
  server3.addHandler(&ws);

  server3.begin();
  /********************END FROM OCV_COLORTRACK11_MS_34_3************/

  // Start streaming web server
  startCameraServer();
}

//ANN 13
void loop() {
  delay(1);

  /********************FROM OCV_COLORTRACK11_MS_34_3************/
  if(One_Time_Transmit==1){
     delay(4000);
     initializeTT();
     i2cTransmit();
     One_Time_Transmit = 0;
  }
   /********************END FROM OCV_COLORTRACK11_MS_34_3************/
}
```

PYTHON STREAMING WINDOW PROGRAM DESCRIPTION

Some of the code, particularly **ANN:5** below, in this program is from: https://github.com/Dallyla/delorean-express

#ANN:1 Shows the imports for this program including FILE_2_C_15M which is the module facilitating a simplex communication link to the Xbox Controller. Immediately below this import are the lists which hold SEND and RCVD data related to the link. The state of the system is determined by the `list_state`.

#ANN:2 defines a text display function. This display shows data received from the Xbox Controller; button-press and joystick positions.

#ANN:3 `Combo_Comm` function performs a one-shot initialization of the simplex channel between the Window program and the Controller. This simplex channel allows the Window to send data to the Controller and receive data from the Controller. Following the initialization, the module sends data (zeros after the initialization) to the Window. Then the SEND list is loaded with results of the OpenCV routine (in this case, centroid data) and sent to the channel. Following that, channel data is received from the channel (Xbox button and joystick data) and placed into the RCVD list.

ANN:3C shows the `Combo_Comm` call, which follows the video display.

Also shown is the need to press the ESC key before closing the program with the X in the upper right corner of the streaming window.

#ANN:4 opens communication with the ESP32 server. See Robot Streaming Server **ANN:11,12**.

#ANN:5 is the beginning of the program and is called from main. a and b are the beginning and end of the bit stream, bts, of the jpg file being transmitted from the server. The img is formed from the jpg.

#ANN:6 The BGR image is converted to HSV and the RED, BLUE, and GREEN ranges for Hue, Saturation, and Value are defined. Note that if the `list_state[1] = 1` then the range for RED, BLUE, or Green is selected from the RCVD list, i.e. button—and button—on the Xbox controller.

#ANN:7 Tracks the selected color by creating a binary mask, delineating all contours of the mask, and finding the largest area contour.

#ANN:8 The center of the largest contour, if there is a contour, is found and this center is also found relative to the center of the screen (`xTrack`, `yTrack`). `xTtack` and `yTrack` are scaled so they fall in the range -128 to +128 for transmission to the Controller program and then to the ESP32 and the MEGA as described in previous chapters.

Also seen here is a crop and resize routine controlled by the Joystick 2Y position transmitted from the Xbox. The crop selects the central portion of the screen and expands it. This results in a ZOOM feature that can be useful for viewing a distant target.

#ANN:9 shows the display of the video with the targeted selected color outlined with a rectangle or a display of the mask depending on `list_state[0]` and `RCVD[2]`,i.e. Controller button X. Also shown is the ZOOM feature.

A table of the above-mentioned features is as follows.

List0[4] == 1 -→ PMODE =1

X ==1 AND Y==0 → OCV MASK

X ==1 AND Y==0

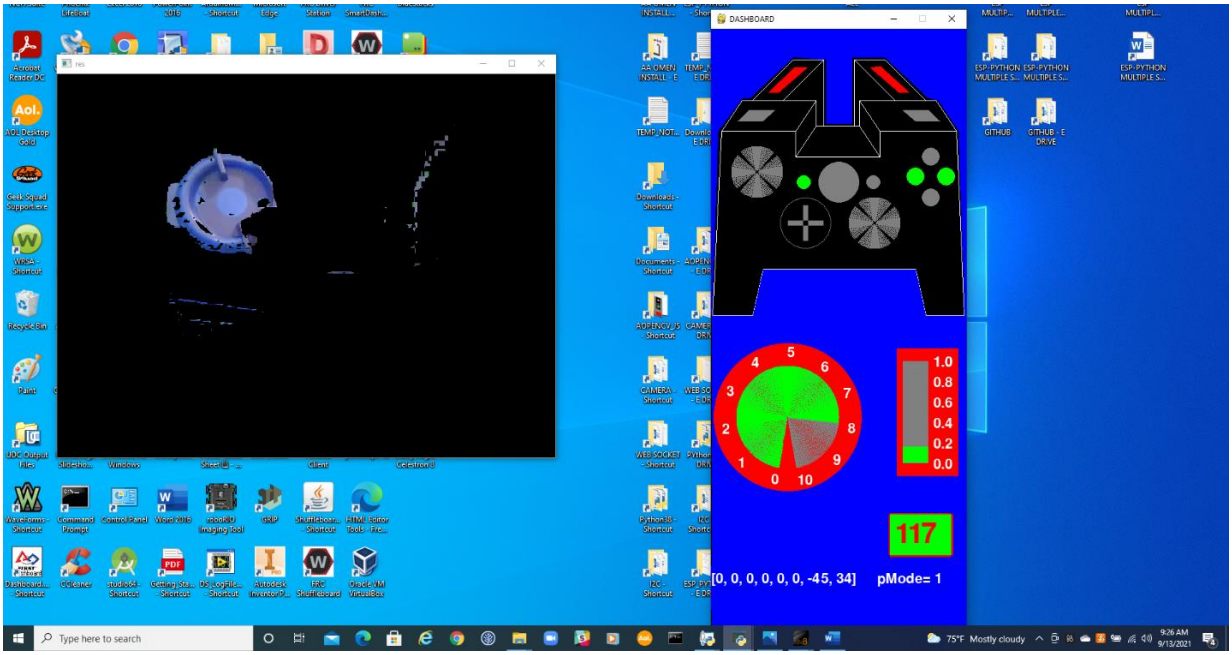
A==0 OR A==1 AND B==0 → RED TRACKING

A==0 AND B== 1 → BLUE TRACKING

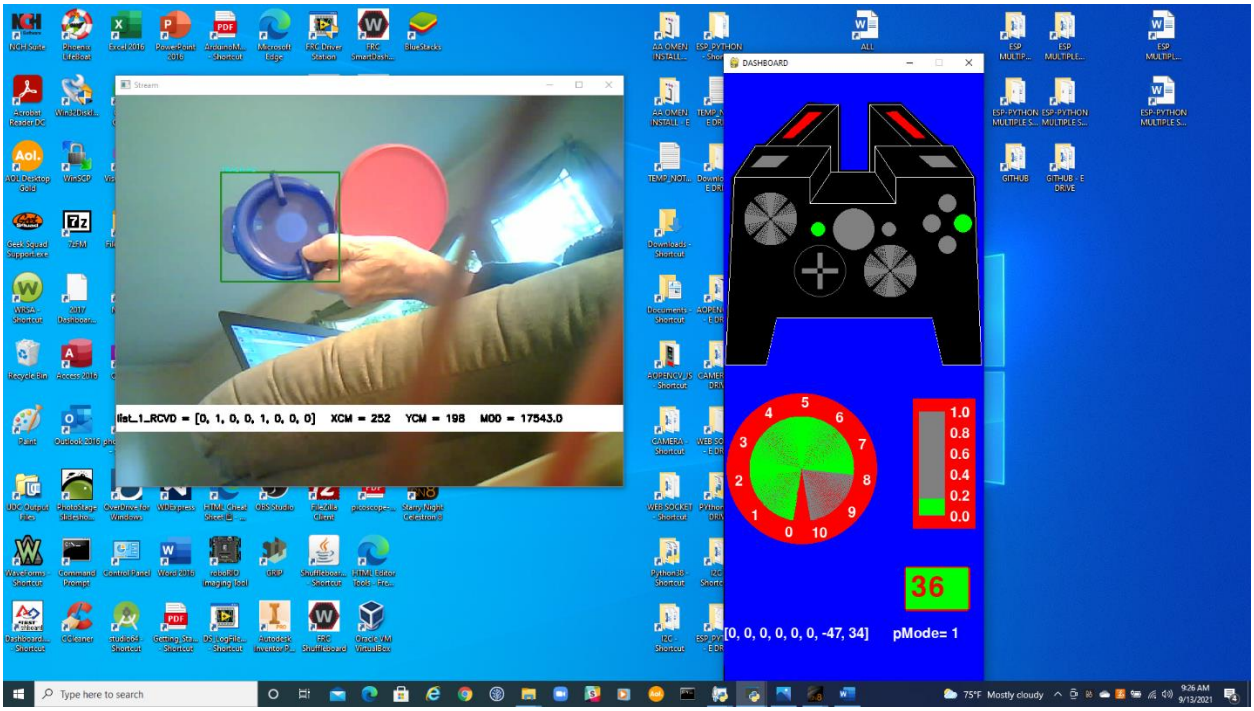
A==1 AND B==1 - → GREEN TRACKING

X==0 Y==1 -> ZOOM

In the screenshot below, the portion of the scene not BLUE is masked out as controlled by the XBOX Controller.



In the screenshot below, as dictated by the XBOX Controller, the Blue object is framed by the green rectangle whose centroid coordinates are communicated to the XBOX Controller program and then back to the ESP32 and then to the Robot main processor (the Arduino MEGA).



PYTHON STREAMING SYSTEM PROGRAM CODE

ROBOT_STREAMING_CLIENT_P.py

You can download the code at the following link:

- https://github.com/arshacker/ESP32-ROBOT-CAM-VIDEO-XBOX-CONTROLLER-OPENCV-SYSTEM/blob/main/ROBOT_STREAMING_CLIENT_P.py

```
########################################################################
#THE VIDEO STREAMING ACQUISITION INSTRUCTIONS AND ASSOCIATED CODE WERE
#FOUND IN GITHUB, DELOREAN-EXPRESS
# https://github.com/Dallyla/delorean-express
########################################################################
#ANN:1
import cv2
from urllib import request
import numpy as np
import time
import random
import GFILE_2_C_15M

###############COMBO COMM
one_shot = 1
list_2_SEND = [0,0,0,0,0,0,0,0]
list_1_RCVD = [0,0,0,0,0,0,0,0]
xcm = 0
ycm = 0
xTrack = 0
yTrack = 0

list_state = [1,1,0,0,0,0,0,0]
              #MASK VIEW IS 1
                #COLOR:0 RED, 1 BLUE 2 GREEN

s_destroyOnce = 0
m_destroyOnce = 0
p_destroyOnce = 0
q_destroyOnce = 0

def probe_point(img,x,y):
    center = (x,y)
    radius = 10
    circle_color = (0,0,0)
    circle_width = -1

    cv2.circle(img,
    center,
    radius,
```

```
    circle_color,
    circle_width)

#ANN:2
def text_display(img,text):
    font = cv2.FONT_HERSHEY_SIMPLEX
    bottomLeftCornerOfText = (1,500)
    fontScale = 0.5
    fontColor = (0,0,0)
    lineType = 2

    topCornerXY = (0,475)
    bottomCornerXY = (800,515)   #image is approx 800X600
    color = (255,255,255)
    lineWidth = -1


    cv2.rectangle(img,
        topCornerXY,
        bottomCornerXY,
        color,
        lineWidth)


    cv2.putText(img,text,
        bottomLeftCornerOfText,
        font,
        fontScale,
        fontColor,
        lineType)

#ANN:3
def Combo_Comm():
    global one_shot
    global list_1_RCVD
    global xcm
    global ycm
    global xTrack
    global yTrack

    if (one_shot ==1):
        GFILE_2_C_15M.init_OneFile()
        one_shot = 0
    startTime = time.time()
    print('list_1_RCVD =',list_1_RCVD)
    print('xTrack_DEBUG = ',xTrack)
    GFILE_2_C_15M.storeToTwoFile()
    for j in range(8):
         #list_2_SEND[j] = list_2_SEND[j] + 2
         #list_2_SEND[7] = 66
         #list_2_SEND[6] = round(xcm)
         #list_2_SEND[7] = round(ycm)
         list_2_SEND[6] = round(xTrack)
         list_2_SEND[7] = round(yTrack)
```

```
        #FILE_2_C_15M.db['DATA_2'][FILE_2_C_15M.name_list_2[j]] = list_2_SEND[j]
        GFILE_2_C_15M.list_2_SEND[j] = list_2_SEND[j]
    GFILE_2_C_15M.accessFromOneFile()
    print('list_1_RCVD_0 =',GFILE_2_C_15M.list_1_RCVD)
    list_1_RCVD=GFILE_2_C_15M.list_1_RCVD
    print('list_1_RCVD =',list_1_RCVD)
    endTime = time.time()
    print('TIME DURATION ===>',endTime-startTime)
    #time.sleep(.5)
###############END COMBO COMM

font = cv2.FONT_HERSHEY_SIMPLEX

current_milli_time = lambda: int(round(time.time() * 1000))

#ANN:4
#stream = request.urlopen('http://192.168.1.10/stream')
#stream = request.urlopen('http://192.168.1.10/stream_handler')
stream = request.urlopen('http://X.X.X.X') #SERVER IP ADDRESS
#stream = cv2.VideoCapture(0)

bts = b''
count = 0
total_fails = 10
starttime = time.time()
t1 = current_milli_time()
t2 = t1

lower_range= np.array([110,50,50], dtype=np.uint8)
upper_range = np.array([130,255,255], dtype=np.uint8)

#ANN:5
def do_run():
  global bts
  global count
  global total_fails
  global starttime
  global t1
  global t2
  global xcm
  global ycm
  global xTrack
  global yTrack
  global s_destroyOnce
  global lower_range
  global upper_range
  global p_destroyOnce

  while True:

    #Combo_Comm()
    bts += stream.read(1024)
    a = bts.find(b'\xff\xd8')
    b = bts.find(b'\xff\xd9')
```

```
    #print(a, b)
    #Combo_Comm()
    if a != -1 and b != -1:
        jpg = bts[a:b+2]
        bts = bts[b+2:]  #cant replace nothing to right of colon with 0
        #img = cv2.imdecode(np.fromstring(jpg, dtype=np.uint8), cv2.IMREAD_COLOR)
        img = cv2.imdecode(np.frombuffer(jpg, dtype=np.uint8), cv2.IMREAD_COLOR)
        # get image height, width
        (width, height) = img.shape[0:2]  #put in 0 instead of nothing to left of
colon

        width=round(1.3*width)     #width = 780
        height=round(0.75*height)  #height = 600
        #print("HEIGHT = ")
        #print(height)
        # calculate the center of the image
        center = (height / 2, width / 2)

        angle0 = 0
        #angle90 = 90
        #angle180 = 180
        #angle270 = 270    #ORIGINAL

        scale = 1.0


        M = cv2.getRotationMatrix2D(center, angle0, scale)
        rotated_0 = cv2.warpAffine(img, M, (width, height))


        # cv2.imshow('Video', img)

        #ANN:6
        # convert BGR image to a HSV image
        hsv = cv2.cvtColor(rotated_0, cv2.COLOR_BGR2HSV)

        # NumPy to create arrays to hold lower and upper range
        # The “dtype = np.uint8” means that data type is an 8 bit integer [36, 28,
75]

        #RED
        #if list_state[1] == 0:
         if list_1_RCVD[4] == 1 and (list_1_RCVD[0] == 0 or list_1_RCVD[0] == 1)
and list_1_RCVD[1] == 0:
            lower_range = np.array([136,87,111], dtype=np.uint8)
            upper_range = np.array([180,255,255], dtype=np.uint8)
        #                           H   S   V

        #BLUE from another tutorial for hsv
        if list_1_RCVD[4] == 1 and list_1_RCVD[0] == 0 and list_1_RCVD[1] == 1:
            lower_range = np.array([110,50,50], dtype=np.uint8)
            upper_range = np.array([130,255,255], dtype=np.uint8)

        #GREEN from opencv color wheel for hsv
```

```
        if list_1_RCVD[4] == 1 and list_1_RCVD[0] == 1 and list_1_RCVD[1] == 1:
            lower_range = np.array([35,50,50], dtype=np.uint8)
            upper_range = np.array([80,255,255], dtype=np.uint8)

        #ANN:7
        # create a mask for image
        if list_1_RCVD[4] == 1:
            mask = cv2.inRange(hsv, lower_range, upper_range)


            #Tracking the  Color
                    (contours,  hierarchy)=cv2.findContours(mask,  cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)


            for pic, contour in enumerate(contours):
                t1 = current_milli_time()
                area = cv2.contourArea(contour)
                if(area>300):
                    x,y,w,h = cv2.boundingRect(contour)
                                                                    rotated_0     =
cv2.rectangle(rotated_0,(x,y),(x+w,y+h),(35,142,35),2)
                    cv2.putText(rotated_0,"TRACKING",(x,y),cv2.FONT_HERSHEY_PLAIN,
0.7, (255,255,0))


                    if count < total_fails and (t1-t2) >= 500:
                        count += 1
                        print(count)
                        falhas = 'err-%s.png' %(count)
                        latitude = random.random() * random.randint(10,50)
                        longitude = random.random() * random.randint(10,50)
                                    cv2.putText(rotated_0,'Coord',(1,30),  font,
0.8,(255,255,255),1)
                          cv2.putText(rotated_0,'X: -{}'.format(latitude),(1,45),
font, 0.5,(255,255,255),1)
                           cv2.putText(rotated_0,'Y: -{}'.format(longitude),(1,60),
font, 0.5,(255,255,255),1)
                        cv2.imwrite(falhas, rotated_0)
                        t2 = t1
        #ANN:8
            try:
                xcm = 0
                ycm = 0
                xTrack = 0
                yTrack = 0
                areas = [cv2.contourArea(temp) for temp in contours]
                max_index = np.argmax(areas)
                #largest_contour = contours[max_index]
                Moments = cv2.moments(contours[max_index])
                M00 = Moments['m00']
                if M00 > 525:  # want a min size target
                    M10 = Moments['m10']
                    M01 = Moments['m01']
```

```
                    xcm = M10/M00
                    ycm = M01/M00
                    xTrack = (xcm - (width/2))/3
                    yTrack = -(ycm - (height/2))/3
                    print('XCM === ',xcm)
                    print('YCM === ',ycm)
                    print('XTRACK == ',xTrack)
                    print('YTRACK == ',yTrack)
                   text_display(rotated_0,'list_1_RCVD = ' + str(list_1_RCVD) + \
                       '   XCM = ' + str(round(xcm)) + '   YCM = ' + \
                       str(round(ycm)) + '   M00 = ' + str(M00))
               else:
                text_display(rotated_0,'list_1_RCVD = ' + str(list_1_RCVD) + '    TARGET TOO SMALL')
           except:
                print('ERROR')
       else:
           print('NO TRACKING')
           text_display(rotated_0,'list_1_RCVD = ' + str(list_1_RCVD))

       #ANN:8A
       #CROP####################

        cropped = rotated_0[int(0.25*height*(abs(list_1_RCVD[7])/127 )):int(height-
0.25*height*(abs(list_1_RCVD[7])/127 )),\
                            int(0.25*width*(abs(list_1_RCVD[7])/127 )):int(width-
0.25*width*(abs(list_1_RCVD[7])/127 ))]

       (width_cr, height_cr) = cropped.shape[0:2]

       #cropped = rotated_0[200:400,200:400]
       #END CROP################

       #RESIZE###############

       width_new = round(width*(1+1.0*(abs(list_1_RCVD[7])/127 )))
       height_new = round(height*(1+1.0*(abs(list_1_RCVD[7])/127 )))
       dim_new = (width_new,height_new)
       #resized = cv2.resize(rotated_0,dim_new,interpolation = cv2.INTER_AREA)
       resized = cv2.resize(cropped,dim_new,interpolation = cv2.INTER_AREA)


           #cv2.destroyAllWindows()
           #s_destroyOnce = 0
           #cv2.imshow('Stream',resized)
       #END RESIZE###############


       #ANN:9
       if list_1_RCVD[4] >= 0 and list_1_RCVD[2] == 0 and list_1_RCVD[3] == 0:
           if s_destroyOnce == 0:
               cv2.destroyAllWindows()
               s_destroyOnce = 1
               m_destroyOnce = 0
               p_destroyOnce = 0
               q_destroyOnce = 0
```

```
            cv2.imshow('Stream',rotated_0)
            #cv2.imshow('Stream',resized)
            #cv2.imshow('Stream',r)
        if list_1_RCVD[4] >= 0 and list_1_RCVD[2] == 0 and list_1_RCVD[3] == 1:
            if p_destroyOnce == 0:
                cv2.destroyAllWindows()
                p_destroyOnce = 1
                s_destroyOnce = 0
                q_destroyOnce = 0
            cv2.imshow('Streamer',resized)
            #cv2.imshow('Streamer',cropped)
        if list_1_RCVD[4] == 1 and list_1_RCVD[2] == 1 and list_1_RCVD[3] == 0:
            # Bitwise-AND mask and original image->show masked image
            res = cv2.bitwise_and(rotated_0,rotated_0, mask= mask)
            print('RES RES VALUE = ',res[225,450])
            if m_destroyOnce == 0:
                cv2.destroyAllWindows()
                m_destroyOnce = 1
                s_destroyOnce = 0
                p_destroyOnce = 0
                q_destroyOnce = 0
            cv2.imshow('res',res)
        if list_1_RCVD[4] == 1 and list_1_RCVD[2] == 1 and list_1_RCVD[3] == 1:
            print("UNDEFINED")
            if q_destroyOnce == 0:
                cv2.destroyAllWindows()
                q_destroyOnce = 1
                m_destroyOnce = 0
                p_destroyOnce = 0
                s_destroyOnce = 0
            cv2.imshow('Stream',rotated_0)

        #ANN:3C
        Combo_Comm()

         if cv2.waitKey(1) == 27: #esc key ends loop. now can hit X on stream
window
                break


def main():
  do_run()

if __name__=="__main__":
  main()
```

PYTHON STREAMING SYSTEM CONTROLLER PROGRAM DESCRIPTION

As mentioned in a previous chapter, the Controller program for the Python Streaming System can communicate with the Python client program as will be described below. This capability can be used in the Teleop or Autonomous Mode. In the Teleop Mode it can be used for basic video control while in the Autonomous Mode it can be used for programming alternative OpenCV options, THEREBY REPLACING TRADITIONAL TRACKBARS—WHICH ARE THE ONLY OPENCV USER CONTROL TOOLS.

The Xbox Controller program for the Browser System, depends on 2 critical imports shown at ANN:1; `pygame.locals`, `dashboard_26_module`, and `GFILE_1_C_15M.PY` module, as well as several generic imports.

`GFILE_1_C_15M.PY` module used to communicate data from the Controller to the Window streaming video and receive data from the Window. This two-way communication is applied when the system is in the Autonomous Robot mode and the video is being analyzed using OpenCV. The module is described below.

ANN:2 shows the web socket address of the ESP32 server with which the Controller will communicate two-way. See Robot Streaming Server **ANN:11,12**.

ANN:2C shows lists that contain SEND and RCVD data. Each list element will be limited to a numeric between -128 to +128. Following the list declarations, the `Combo_Comm` function performs a one-shot initialization of the simplex channel between the Controller and the Window program. This simplex channel allows the Controller to send data to the Window and receive data from the Window. Following the initialization, the module sends data (zeros after the initialization) to the Window. Then the SEND list is loaded with joystick and button data of the Autonomous Command button, list0[7] ==1, and sent to the channel. Following that, channel data is received from the channel and placed into the RCVD list. **ANN:2C1** shows the `Combo_Comm` call, which follows the collection of all Controller inputs. All Controller data, except that replaced by RCVD data, is placed in listAuton. See **ANN:5C**, within **ANN: 5**, for the display of RCVD data and **ANN:9** for further action for `listAuton`.

#ANN:3 are module constants and function lists used to call the GUI components.

#ANN:4 is list0 showing the numeric state of each element (button, trigger, etc) of the Controller. The ID number of each element of the list corresponds to the Diagram—except for the PULSE whose purpose will be discussed below.

#ANN:5 shows the function which updates the Dashboard on the GUI using list0 as described immediately above.

#ANN:6 Is the principal program called from main.

#ANN:7 shows the "FILL" which resets the colors of particular elements to prepare for the next redraw of the Controller elements. Immediately follow the `FILL` instructions is `list0`[19] which serves as the "pulse" of the Controller which displays a number continuously increasing by 1, transmitted to the ESP32 and then back to the Controller and displayed. This display assures the operator that the communication link is operational.

#ANN:8 shows the collection of all Controller inputs, latching all that are designated as latched inputs, and inserting them into list0. (remember **ANN:5** above). Note that the list0[6] element corresponding to L1 button is coded to have 4 states--0,1,2,3—as described above, and can be expanded to more states, easily.

#ANN:9 shows transmission of list0 or listAuton to the ESP32 server depending on the state of the Programming Button, `list0[6]`.

#ANN:10 shows the reception of sensor data from the server. (The sensor data is entered into a list array referenced in the dashboard update described in **ANN:5**.) To completely understand the sensor data operation of ACKDATA1, the reader is referred to The Zero Hack section of this paper.

XBOX CONTROLLER CODE

ROBOT_STREAMING_XBOX_P.py

You can download the code at the following link:

- https://github.com/arshacker/ESP32-ROBOT-CAM-VIDEO-XBOX-CONTROLLER-OPENCV-SYSTEM/blob/main/ROBOT_STREAMING_XBOX_P.py

```
#ANN:1
import pygame
from pygame.locals import *
import dashboard_26_module
import time
import datetime
import math
import keyboard
import GFILE_1_C_15M

wsComm = 1

#ANN:2
import websocket  #actually websocket-client; websocket gives wrong WebSocket()

if wsComm == 1:
    ws = websocket.WebSocket()
    ws.connect("ws://X.X.X.X:82/ws") #ROBOT STREAMING SERVER IP ADDRESS

##############COMBO COMM
one_shot = 1
#ANN:2C
list_1_SEND = [0,0,0,0,0,0,0,0]
list_2_RCVD = [0,0,0,0,0,0,0,0]

def Combo_Comm():
    global one_shot
    global list_2_RCVD
    if (one_shot ==1):
        GFILE_1_C_15M.init_TwoFile()
        one_shot = 0
    #print('list_2_RCVD =',list_2_RCVD)
    GFILE_1_C_15M.storeToOneFile()
    #db['DATA_1']['list_1[0]'] = db['DATA_1']['list_1[0]'] + 1
    for j in range(8):
        list_1_SEND[7] = list0[12]
        list_1_SEND[6] = list0[11]
        #list_1_SEND[5] = list0[5]
        list_1_SEND[5] = list0[7]
```

```
        list_1_SEND[4] = listAuton[6]
        list_1_SEND[3] = list0[3]
        list_1_SEND[2] = list0[2]
        list_1_SEND[1] = list0[1]
        list_1_SEND[0] = list0[0]
                    #FILE_1_C_15M.db['DATA_1'][FILE_1_C_15M.name_list_1[j]]    =
list_1_SEND[j]
        GFILE_1_C_15M.list_1_SEND[j] = list_1_SEND[j]
    GFILE_1_C_15M.accessFromTwoFile()
    #list_2_RCVD = GFILE_1_C_15M.list_2_RCVD
    print('list_2_RCVD_0 =',GFILE_1_C_15M.list_2_RCVD)
    list_2_RCVD=GFILE_1_C_15M.list_2_RCVD
    print('list_2_RCVD =',list_2_RCVD)
    #time.sleep(.1)


################END COMBO COMM

i = 0

listBIN = [0,0,0,0,0,0,0]

def DecToBin(num,i):

    if num > 1:
        DecToBin(num // 2,i+1) #push down stack, first in-last out
    #print("num = ",num)
    #print("num%2=",num % 2)
    #print("i=",i)
    listBIN[6-i]=num % 2  #6-i reverses so last out goes into listBIN[0]
    #print(listBIN)



index = 0

BYTE_NUMB = 127
#ANN:3
JStick_Arc_Radius = dashboard_26_module.radius_1

listFunc = [dashboard_26_module.Button_A,dashboard_26_module.Button_B,\
            dashboard_26_module.Button_X,dashboard_26_module.Button_Y,\
            dashboard_26_module.Button_L,dashboard_26_module.Button_R,\
            dashboard_26_module.Button_L1,dashboard_26_module.Button_R1,\
            0,0,0,\
            dashboard_26_module.JStick_2X,dashboard_26_module.JStick_2Y,\
            dashboard_26_module.TrigL_Sig,\
            dashboard_26_module.JStick_1X,dashboard_26_module.JStick_1Y,\
            dashboard_26_module.TrigR_Sig,\
            dashboard_26_module.HatX,dashboard_26_module.HatY]

listSensorFunc = [dashboard_26_module.SENSOR_1_VALUE,\
                  dashboard_26_module.SENSOR_2_VALUE,\
                  dashboard_26_module.SENSOR_3_VALUE,\
```

```
                  dashboard_26_module.SENSOR_N_VALUE]
#ANN:4
list0 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
#list0 = [0,1,1,0,1,0,1,1,0,0,0,117,-122,1,127,-127,1,-1,1,0]
#        0 1 2 3 4 5 6 7 X X X 11  12  13   14  15  16    17 18  19
#                              J2X J2Y LTRG J1X J1Y RTRG  HX HY  PULSE
#                               + -
#                              0-127

                #ascii [=91  ]=93  ,=44  sp=32  -=45  0=48

listAuton = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

listACK1 = [0,0,0,0,0]
listACK2 = [0,0,0,0]

list_rcv = [0,0,0,0,0,0,0,0,0,0]

listACK_rcv = [0,0,0,0,0,0,0,0,0,0]

listACK_rcv_ord = [0,0,0,0,0,0,0,0,0,0]  #only first 4 locs used

button_out = [0,0,0,0,0,0,0,0,0,0,0]
button_out_prev = [0,0,0,0,0,0,0,0,0,0,0]
button_prev_zero = 0
button_out_diff = [0,0,0,0,0,0,0,0,0,0,0]

list0_prev = [0,0,0,0,0,0,0,0,0,0,0]
list0_diff = [0,0,0,0,0,0,0,0,0,0,0]

myCount = 0
COUNTER = 0
MAXCOUNT = 4

incr = +1

running = True

black = (0,0,0)
gray = (128,128,128)
light_gray = (200,200,200)
dark_gray = (64,64,64)
white = (255,255,255)
red = (255,0,0)
light_red =(128,0,0)
green = (0,255,0)
blue = (0,0,255)
listColor_in=[0,0,0]
listColor_in[0]=abs(-255)
Color_in = tuple(listColor_in)

Max_Arc_Width = math.floor((dashboard_26_module.ScreenWidth/250)*25) #25
Max_JStick_Amplitude = 127
#ANN:5
```

```
def update_dashboard():
    for i in [0,1,2,3,4,5,6,7,11,12,13,14,15,16,17,18]:
        if(i<8):
            if(list0[i]==1):
               listFunc[i](green)
            if(list0[i]==0):
               listFunc[i](gray)
        if(i==13 or i == 16):
            #MAPPING FOR NEG X   RED=127+ABS(X)  GREEN=127-ABS(X)
            #        FOR POS X   RED=127-ABS(X)  GREEN=127+ABS(X)
            if(list0[i]<0):
               listFunc[i]((127+abs(list0[i]),127-abs(list0[i]),0))
            if(list0[i]>=0):
               listFunc[i]((127-abs(list0[i]),127+abs(list0[i]),0))
        if(i==11 or i==12 or i==14 or i==15):
            if(list0[i]<0):
               #ARC WIDTH = FRACTIONAL (MAX AMPL-AMPL)/MAX AMPL
               listFunc[i](math.floor(((Max_JStick_Amplitude-abs(list0[i]))/\
                 Max_JStick_Amplitude)*Max_Arc_Width),Max_Arc_Width)
            if(list0[i]>=0):
               #ARC WIDTH = FRACTIONAL (MAX AMPL-AMPL)/MAX AMPL
               listFunc[i](Max_Arc_Width,\
                 math.floor(((Max_JStick_Amplitude-abs(list0[i]))/\
                 Max_JStick_Amplitude)*Max_Arc_Width))
        if(i==17 or i==18):
            if(list0[17]==1 and list0[18]==1):
               listFunc[17](gray,green)
               listFunc[18](green,gray)
            if(list0[17]==1 and list0[18]==0):
               listFunc[17](gray,green)
               listFunc[18](gray,gray)
            if(list0[17]==1 and list0[18]==-1):
               listFunc[17](gray,green)
               listFunc[18](gray,red)
            if(list0[17]==0 and list0[18]==1):
               listFunc[17](gray,gray)
               listFunc[18](green,gray)
            if(list0[17]==0 and list0[18]==0):
               listFunc[17](gray,gray)
               listFunc[18](gray,gray)
            if(list0[17]==0 and list0[18]==-1):
               listFunc[17](gray,gray)
               listFunc[18](gray,red)
            if(list0[17]==-1 and list0[18]==1):
               listFunc[17](red,gray)
               listFunc[18](green,gray)
            if(list0[17]==-1 and list0[18]==0):
               listFunc[17](red,gray)
               listFunc[18](gray,gray)
            if(list0[17]==-1 and list0[18]==-1):
               listFunc[17](red,gray)
               listFunc[18](gray,red)
    for j in [0,1,2,3]:
            if (j==0):
```

```
                listSensorFunc[j](green,listACK2[j]/127)
            if (j==1):
                listSensorFunc[j](green,listACK2[j]/127)
            if (j==2):
                listSensorFunc[j](red,listACK2[j])
            if (j==3):
                listSensorFunc[j](white,list_2_RCVD,0,850)
                listSensorFunc[j](white,'pMode=',260,850)
                listSensorFunc[j](white,myCount,350,850) #ANN:5C

#ANN:6
def do_running():
  global i #error generated in combo version
  global running
  global index
  global listACK1
  global myCount
  global COUNTER
  global incr
  global listAuton
  while (running):
    start_time = datetime.datetime.now()
    #time.sleep(1)
    #ANN:7
    dashboard_26_module.JStick_1(red,green,red,green)
    dashboard_26_module.JStick_2(red,green,red,green)
    dashboard_26_module.SENSOR_1_FILL(gray,-80,260)
    dashboard_26_module.SENSOR_2_FILL(gray)
    dashboard_26_module.SENSOR_3_FILL(green)
    dashboard_26_module.SENSOR_N_FILL(blue)
####################FROM XBOX_TW0_TEST_8_EXPT2############
    list0[19] = index    #pulse of the transmission sytem
    index = index % BYTE_NUMB
    index = index + 1
    #print("List0 = ",list0)
    #ANN:8
  ###GET CONTROLLER INPUTS#################
   # Get count of joysticks
    joystick_count = pygame.joystick.get_count()
    #print(joystick_count)
    for i in range(joystick_count):
        joystick = pygame.joystick.Joystick(i)
        joystick.init()

        # Get the name from the OS for the controller/joystick
        name = joystick.get_name()

       # Usually axis run in pairs, up/down for one, and left/right for the other.
        axes = joystick.get_numaxes()
        #print(axes)
        for j in range( axes ):
            axis = joystick.get_axis( j )
            list0[j+11] = round(BYTE_NUMB*axis)
```

```
        buttons = joystick.get_numbuttons()

        for i in range( buttons ):
            button_out[i] = joystick.get_button( i )
            if(button_out[i]==1 and button_out_prev[i]==0):
                button_out_diff[i] =1
                #COUNTER = COUNTER + 1
            else:
                button_out_diff[i] =0
            #if button_out_diff[6] == 1:
                #COUNTER = COUNTER + 1
            button_out_prev[i] = button_out[i]
            list0[i] = button_out_diff[i]^list0[i]
            if list0[i] == 1 and list0_prev[i]==0:
                list0_diff[i] = 1
            else:
                list0_diff[i] = 0
            list0_prev[i] = list0[i]
        #if button_out_diff[7] == 1:
         #   COUNTER = COUNTER + 0.5
        if myCount == MAXCOUNT and incr == +1:
            incr = -1
        if myCount == 0 and incr == -1:
            incr = +1
        if list0_diff[6] == 1:
            myCount = myCount + incr


        # Hat switch. All or nothing for direction, not like joysticks.
        # Value comes back in a tuple.
        hats = joystick.get_numhats()

        for i in range( hats ):
            hat = joystick.get_hat(i )
            mytuple = hat
        list0[17] = mytuple[0]
        list0[18] = mytuple[1]


    print("list0 = ",list0)

######################END FROM XBOX_TW0_TEST_8_EXPT2########
#ANN:2C1
    Combo_Comm()
    #######FILL LISTAUTON############
    for j in range(20):
        listAuton[j] = list0[j]
        if j == 6:
            listAuton[j] = myCount
    for j in range(11,19):
        listAuton[j] = list_2_RCVD[j-11]
        if j==18:
            print('XXXXXXXXXXXX=',list_2_RCVD[j-11])
    print("listAUTON = ",listAuton)
```

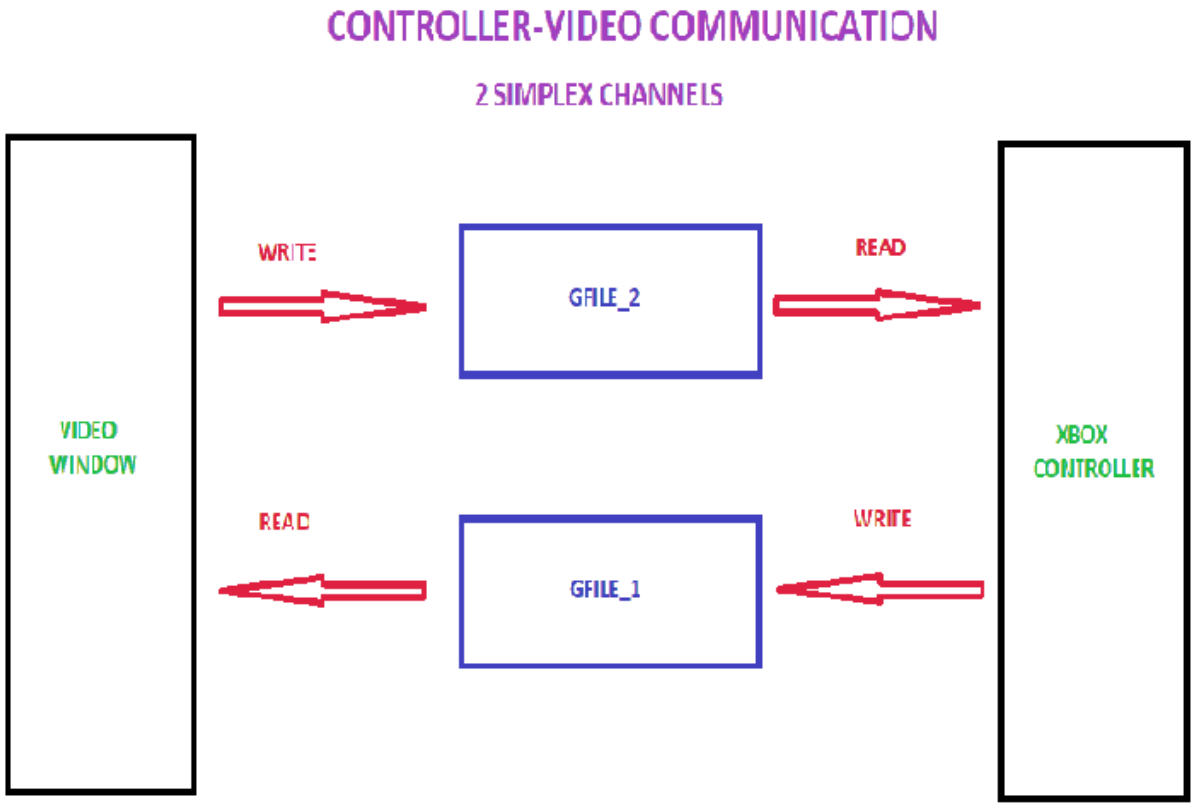
```
    #####END FILL LISTAUTON
    update_dashboard()
    pygame.display.update()
    #Combo_Comm()
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            print("QUIT")
            running = False
            pygame.quit()

###############PREPARE AND SEND MESSAGE#######################
    #ANN:9
    #if list0[6] == 0:  #button L1 TELEOP
    if listAuton[6] == 0:
        string0 = str(list0)
        string0_stripped = string0[1:len(string0)-1] #strip [ and ]
        message = string0_stripped.encode()
    #if list0[6] == 1:  #button L1 AUTON
    if listAuton[6] >= 1:
        print('RUN AUTON == ',listAuton)
        stringA = str(listAuton)
        stringA_stripped = stringA[1:len(stringA)-1] #strip [ and ]
        message = stringA_stripped.encode()
    #if list0[7] == 0: #button R1 AUTON PROGRAMMING OFF
    print("message =",message)
    #message = bytes(list1)
    #ws.send("hello") #works. sends each char as ascii
    if wsComm == 1:
        ws.send(message)   #workd. sends each nume ral as ascii
    #ws.send(5) #does not work
    #ws.send_binary(100|35|50)  #does not work
    #end_time1 = datetime.datetime.now()
    #exec_time1 = end_time1 - start_time
    #print("execTime1 = ",exec_time1)
    index1 = 0
    data = b""
    msgLength = len(message)
    print("msgLength = ",msgLength)

#############END PREPARE AND SEND MESSAGE#######################

#----------ALT RCV MSG-------------------------------
    for x in range(7):
        listBIN[x] = 0
    print("listBIN = ",listBIN)

    for x in range(5):
        listACK1[x] = 0
        #listACK2[x] = 1
    print("listACK1 = ",listACK1)

    for x in range(4):
        listACK2[x] = 1
    print("listACK2 = ",listACK2)
```

```
    #for z in range(10):
        #listACK_rcv[z] = 250
    #print("listACK_rcv = ",listACK_rcv)


    #ANN:10
    if wsComm == 1:
        ACKDATA1 = ws.recv()
    else:
        ACKDATA1 = b'e\x140\x05\x01'  #simulated ws.recv()
    print("ACKDATA1 = ",ACKDATA1)
    listACK1 = list(ACKDATA1)
    #print(list(ACKDATA1))
    print("listACK1 = ",listACK1)
    DecToBin(listACK1[4],i)
    print("DECTOBIN = ",listBIN)
    if listBIN[0]==1:
        listACK2[0]=0
    else:
        listACK2[0]=listACK1[0]
    if listBIN[1]==1:
        listACK2[1]=0
    else:
        listACK2[1]=listACK1[1]
    if listBIN[2]==1:
        listACK2[2]=0
    else:
        listACK2[2]=listACK1[2]
    if listBIN[3]==1:
        listACK2[3]=0
    else:
        listACK2[3]=listACK1[3]
    print("listACK2 = ",listACK2)

    print('MYCOUNT = ',myCount)
    #print('COUNTER = ',COUNTER)

    #print("listACK2 =",listACK2)

    end_time2 = datetime.datetime.now()
    exec_time2 = end_time2 - start_time
    print("execTime2 = ",exec_time2)



#----------END ALT RCV MSG--------------------------

def main():
  dashboard_26_module.Dashboard()

  # Used to manage how fast the screen updates
  #clock = pygame.time.Clock()
```

```
  # Initialize the joysticks
  pygame.joystick.init()

###############COMM  INIT##############################
#import socket
#sock = socket.socket()
#host = "192.168.1.9" #ESP32 IP in local network
#port = 80             #ESP32 Server Port
#sock.connect((host, port))


  #ws = websocket.WebSocket()
  #ws.connect("ws://192.168.1.4:82/ws")

###################END COMM INIT#######################

#######################COMM ESTABLISHED####################

  if wsComm == 1:
      ACKDATA = ws.recv()
  else:
      ACKDATA = "SIMULATED HELLO"
  print("ACKDATA = ",ACKDATA)
  print("PRESS q TO SEND AND z TO CLOSE")

#if(keyboard.is_pressed('q')):

###################END COMM ESTABLISHED####################

  do_running()

if __name__=="__main__":
  main()
```

XBOX CONTROLLER-PYTHON CLIENT COMMUNICATION DESCRIPTION

A diagram showing the communication between the Controller and Video Stream programs is shown below.



The idea is the creation of two files, GFILE_1,2. The Controller program writes data into GFILE_1 and the Video program reads the data from the file. Similarly, the Video Program writes data into GFILE_2 and the Controller program reads the data.

The Controller and Video program run independently so there is a small but finite probability of a “collision”; a read error caused by the

unavailability of a file. A “try-except” allows each program to continue without a significant effect on the user program.

The programs for the GFILE_1_C_15M and GFILE_2_C_15M modules are mirrored images and are described below. The modules are quite simple so annotations will not be used here.

The reader will note the continuation of the use of lists(arrays) in both modules to implement communication between programs. The SEND and RCVD lists are used to interface with the Xbox Controller and Video Streaming programs importing this module.

GFILE_1 MODULE:

- `storetoOneFile` the writing of list_1_SEND data into OneFile.
- `accessFromTwoFile` shows the reading of data from TwoFile, the conversion of the string into a list of data, and the insertion into the list_2_RCVD list.
- `Init_TwoFile` shows the initialization of `TwoFile`.

GFILE_2 MODULE:

- `storetoTwoFile` the writing of list_2_SEND data into TwoFile.
- `accessFromOneFile` shows the reading of data from OneFile, the conversion of the string into a list of data, and the insertion into the `list_1_RCVD` list.
- `Init_OneFile` shows the initialization of OneFile.

XBOX CONTROLLER-PYTHON WINDOW COMMUNICATION CODE

GFILE_1_C_15M.PY

You can download the code at the following link:

- https://github.com/arshacker/ESP32-ROBOT-CAM-VIDEO-XBOX-CONTROLLER-OPENCV-SYSTEM/blob/main/GFILE_1_C_15M.py

```
import cv2
import time

one_shot = 1

list_1_SEND = [0,0,0,0,0,0,0,0]

list_2_RCVD = [0,0,0,0,0,0,0,0]

#list_1 = [0,0,0,0,0,0,0,0]
list_2_INIT  = [0,0,0,0,0,0,0,0]

def storeToOneFile():

    # Its important to use binary mode
    #print("STORETOONE")
    OneFile = open('OneFile', 'w')
    string_list_1_SEND = str(list_1_SEND)
    # source, destination
    OneFile.write(string_list_1_SEND)
    OneFile.close()

def accessFromTwoFile():
    TwoFile = open('TwoFile', 'r')
    try:
        string0 = TwoFile.read()
        print('string0 = ',string0)
        string0_stripped = string0[1:len(string0)-1] #strip [ and ]
        string1 = string0_stripped
        #print(string1)
        string2 = string1.replace(","," ")
        #print(string2)
        listz = string2.split()#STRING INTO LIST OF SUBSTRINGS
        print('listz = ',listz)
        for j in range(8):
           list_2_RCVD[j] = int(listz[j])
           #list_2_RCVD[j] = list_2[j]
```

```
        print('list_2 = ',list_2_RCVD)
        #print(listz[0]-1)
    except:
        print('READ ERROR')
    TwoFile.close()

def init_TwoFile():
    global list_2_RCVD
    TwoFile = open('TwoFile', 'w') #w write
    for i in range(8):
        list_2_INIT[i] = 0
        list_2_RCVD[i] = 0
        list_1_SEND[i] = 0
    TwoFile.write(str(list_2_INIT))
    TwoFile.close()

def do_run():
  while True:
    global one_shot
    if (one_shot ==1):
        init_TwoFile()
        one_shot = 0
    print('list_2_RCVD =',list_2_RCVD)
    storeToOneFile()
    for j in range(8):
         list_1_SEND[j] = list_1_SEND[j] + 10
         #list_1[j] = list_1_SEND[j]
    accessFromTwoFile()
    time.sleep(.3)
    if cv2.waitKey(1) == 27:  #esc key ends loop. now can hit X on stream window
                print('break')
                #break

def main():
    do_run()

if __name__=="__main__":
  main()
import cv2
import time

one_shot = 1

list_1_SEND = [0,0,0,0,0,0,0,0]

list_2_RCVD = [0,0,0,0,0,0,0,0]

#list_1 = [0,0,0,0,0,0,0,0]
list_2_INIT  = [0,0,0,0,0,0,0,0]

def storeToOneFile():

    # Its important to use binary mode
    #print("STORETOONE")
```

```
    OneFile = open('OneFile', 'w')
    string_list_1_SEND = str(list_1_SEND)
    # source, destination
    OneFile.write(string_list_1_SEND)
    OneFile.close()

def accessFromTwoFile():
    TwoFile = open('TwoFile', 'r')
    try:
        string0 = TwoFile.read()
        print('string0 = ',string0)
        string0_stripped = string0[1:len(string0)-1] #strip [ and ]
        string1 = string0_stripped
        #print(string1)
        string2 = string1.replace(","," ")
        #print(string2)
        listz = string2.split()#STRING INTO LIST OF SUBSTRINGS
        print('listz = ',listz)
        for j in range(8):
           list_2_RCVD[j] = int(listz[j])
           #list_2_RCVD[j] = list_2[j]
        print('list_2 = ',list_2_RCVD)
        #print(listz[0]-1)
    except:
        print('READ ERROR')
    TwoFile.close()

def init_TwoFile():
    global list_2_RCVD
    TwoFile = open('TwoFile', 'w') #w write
    for i in range(8):
        list_2_INIT[i] = 0
        list_2_RCVD[i] = 0
        list_1_SEND[i] = 0
    TwoFile.write(str(list_2_INIT))
    TwoFile.close()

def do_run():
  while True:
    global one_shot
    if (one_shot ==1):
        init_TwoFile()
        one_shot = 0
    print('list_2_RCVD =',list_2_RCVD)
    storeToOneFile()
    for j in range(8):
         list_1_SEND[j] = list_1_SEND[j] + 10
         #list_1[j] = list_1_SEND[j]
    accessFromTwoFile()
    time.sleep(.3)
    if cv2.waitKey(1) == 27:  #esc key ends loop. now can hit X on stream window
                print('break')
                #break
```

```
def main():
    do_run()

if __name__=="__main__":
  main()
```

GFILE_2_C_15M.PY

You can download the code at the following link:

- https://github.com/arshacker/ESP32-ROBOT-CAM-VIDEO-XBOX-CONTROLLER-OPENCV-SYSTEM/blob/main/GFILE_2_C_15M.py

```
import cv2
import time

one_shot = 1

list_2_SEND = [0,0,0,0,0,0,0,0]

list_1_RCVD = [0,0,0,0,0,0,0,0]

list_1_INIT = [0,0,0,0,0,0,0,0]
#list_2  = [0,0,0,0,0,0,0,0]

def storeToTwoFile():

    # Its important to use binary mode
    #print("STORETOTWO")
    TwoFile = open('TwoFile', 'w')
    string_list_2_SEND = str(list_2_SEND)
    # source, destination
    TwoFile.write(string_list_2_SEND)
    TwoFile.close()

def accessFromOneFile():
    OneFile = open('OneFile', 'r')
    try:
        string0 = OneFile.read()
        print('string0 = ',string0)
        string0_stripped = string0[1:len(string0)-1] #strip [ and ]
        string1 = string0_stripped
        #print(string1)
        string2 = string1.replace(","," ")
        #print(string2)
        listz = string2.split()#STRING INTO LIST OF SUBSTRINGS
        print('listz = ',listz)
        for j in range(8):
           list_1_RCVD[j] = int(listz[j])
           #list_1_RCVD[j] = list_1[j]
```

```
        #print(listz[0]-1)
    except:
        print('READ ERROR')
    OneFile.close()

def init_OneFile():
    global list_1_RCVD
    OneFile = open('OneFile', 'w') #w write
    for i in range(8):
        list_1_INIT[i] = 0
        list_1_RCVD[i] = 0
        list_2_SEND[i] = 0
    OneFile.write(str(list_1_INIT))
    OneFile.close()

def do_run():
  while True:
    global one_shot
    if (one_shot ==1):
        init_OneFile()
        one_shot = 0
    print('list_1_RCVD =',list_1_RCVD)
    storeToTwoFile()
    for j in range(8):
         list_2_SEND[j] = list_2_SEND[j] + 2
         #list_2[j] = list_2_SEND[j]
    accessFromOneFile()
    time.sleep(.3)
    if cv2.waitKey(1) == 27: #esc key ends loop. now can hit X on stream window
                break

def main():
    do_run()


if __name__=="__main__":
  main()
```

ArduinoRobotMega.ino & ArduinoRobotProMini.ino PROGRAM DESCRIPTION

The *ArduinoRobotMega.ino* and *ArduinoRobotProMini.ino* programs are virtually identical.

The former program is a skeleton program that simply receives control signals from the ESP32 (converting some to signed integers) and returns sensor data to the ESP32. It is left to the reader to fill in the skeleton by converting incoming control signals to actuator commands and supplying sensor signals for any specific robot being designed. It also drives an OLED display using an SPI interface. If the SPI interface is needed for other purposes, an Arduino Pro Mini on the ESP32 I2C bus can be used to drive the display on its SPI interface.

The latter program is furnished for the dedicated micro driving the OLED display using the SPI interface.

The difference between the two programs is their I2C addresses. Although this is a small difference, both programs are listed below in the Code section for the convenience of the reader.

ANN 1 shows the necessary include files for the I2C and SPI interfaces. It shows the buffer and `bufSigned` control data array received by the Arduino Robot micro. It also shows the `TXbuf` array containing sensor data for transmission to the ESP32. As mentioned above, the use of the control data array is left to the reader. Similarly, sensor data is simulated

here with made-up numbers which are transmitted to the ESP32. Actual sensor operations are left to the reader,

ANN 2 shows the interrupt-driven function which transmits sensor data to the ESP32 upon request by the ESP23

ANN 3 shows the interrupt-driven function which receives control data from the ESP32.

ANN OLED shows the display function which displays the IP Address of the ESP32.

The setup function is fairly self-explanatory as is the loop contents.

ArduinoRobotMega.ino CODE

You can download the code at the following link:

- https://github.com/arshacker/ESP32-ROBOT-CAM-VIDEO-XBOX-CONTROLLER-OPENCV-SYSTEM/blob/main/ArduinoRobotMega_ANN/ArduinoRobotMega_ANN.ino

```
// This is the sketch for the Slave Arduino using the hardware I2C.
//ANN 1
#include <Arduino.h>   //oled
#include <U8g2lib.h>   //oled

#include <Wire.h>

#include <SPI.h>   //spi oled

volatile byte buffer[40];
volatile int rxHowMany;
volatile int rxInterrupts = 0;
volatile boolean flagRequest;

signed short int bufSigned[40];
byte TXbuf[] = { 101, 20, 110, 5, 100, 105, 44, 2, 120, 72 }; //ars

int bufInt[40];
int printOnce = 0;
```

```
//char basciibuf[20];

U8G2_SH1106_128X64_NONAME_F_4W_HW_SPI u8g2(U8G2_R0, /* cs=*/ 10, /* dc=*/ 9, /*
reset=*/ 8);  //clk pin 13 UNO mosi pin 11 UNO

//ANN OLED
void ipPrint(void){
  int a = bufSigned[0];
  char asciibuf[20]={0,};
  int b = bufSigned[1];
  int c = bufSigned[2];
  int d = bufSigned[3];

  //char bsciibuf[3];
  itoa(a,asciibuf,10); //10 is decimal
  itoa(b,asciibuf+4,10);
  itoa(c,asciibuf+8,10);
  itoa(d,asciibuf+12,10);

//  itoa(c,basciibuf,10);

  //asciibuf[4]=" ";


  char totalbuf[20];
  totalbuf[0]=asciibuf[0];
  totalbuf[1]=asciibuf[1];
  totalbuf[2]=asciibuf[2];
  totalbuf[3]=asciibuf[4];
  totalbuf[4]=asciibuf[5];
  totalbuf[5]=asciibuf[6];
  totalbuf[6]=asciibuf[7];

  //String myStr1 = String(bufSigned[0]);

  //ug82.drawStr(xcoord=5 x nbr spaces,ycoord=10 x nbr lines,"STRING")

  u8g2.clearBuffer();          // clear the internal memory
  u8g2.setFont(u8g2_font_ncenB08_tr); // choose a suitable font
  u8g2.drawStr(0,10,"IP ADDRESS");  // write 1ST LINE to the internal memory
  u8g2.drawStr(0,20,asciibuf);  // write 2ND LINE to the internal memory
  //u8g2.drawStr(0,20,myStr1);  // write 2ND LINE to the internal memory
  u8g2.drawStr(25,20,asciibuf+4);  // write 2ND LINE to the internal memory
  u8g2.drawStr(50,20,asciibuf+8);  // write 2ND LINE to the internal memory//NEW
  u8g2.drawStr(75,20,asciibuf+12);    //   write   2ND   LINE   to   the   internal
memory//NEW
  //u8g2.drawStr(50,20,asciibuf+4);  // EVEN THIS DOESNT WORK???
  //u8g2.drawStr(50,20,asciibuf+8);  // write 2ND LINE to the internal memory
  //u8g2.drawStr(50,20,"K");
  //u8g2.drawStr(45,20,basciibuf);  // write 2ND LINE to the internal memory
  //u8g2.drawStr(0,30,asciibuf+4);  // write 2ND LINE to the internal memory
  //u8g2.drawStr(0,40,asciibuf+8);  // write 2ND LINE to the internal memory
  //u8g2.drawStr(25,40,asciibuf+12);  // write 2ND LINE to the internal memory
```

```
  //u8g2.drawStr(0,50,asciibuf+12);  // write 2ND LINE to the internal memory
  u8g2.drawStr(0,30,"SIGN IN NOW!");  // write LAST LINE to the internal memory
  u8g2.sendBuffer();          // transfer internal memory to the display
  delay(1000);
}

void testPrint(void){
  u8g2.clearBuffer();          // clear the internal memory
  u8g2.setFont(u8g2_font_ncenB08_tr); // choose a suitable font
  u8g2.drawStr(0,10,"Hello World!");  // write 1ST LINE to the internal memory
  u8g2.drawStr(0,20,"Hello Again World!");  // write 2ND LINE to the internal
memory
  u8g2.drawStr(0,60,"Bye For Now World!");  // write LAST LINE to the internal
memory
  u8g2.sendBuffer();          // transfer internal memory to the display
  delay(1000);
}

//ANN 2
void requestEvent()
{
  static byte x = 0;

  // Fill array with numbers.



  //TXbuf[0] = x++;         // overwrite the first with a counter.
  Wire.write(TXbuf, sizeof(TXbuf));

  flagRequest = true;
}


//ANN 3
void receiveEvent(int howMany)   //HOWMANY BYTES SENT BY MASTER--5 BYTES //ars
{
  for( int i=0; i<howMany; i++)
    buffer[i] = Wire.read();

  rxHowMany = howMany;
  rxInterrupts++;
}

void setup()
{
  Serial.begin(9600);           // start serial for output
  Serial.println("\nSlave 2");

  Wire.begin(6);                // join i2c bus as 2ND slave with address #6
  Wire.onReceive(receiveEvent); // interrupt handler for receiving i2c data
  Wire.onRequest(requestEvent); // interrupt handler for when data is requested by
i2c
  for(int y=0; y<40; y++){
```

```
    bufSigned[y] = 0;
  }
  u8g2.begin();   //oled
  delay(10);
  //testPrint();
}

//ANN 4
void loop()
{
  delay(2000);
  noInterrupts();
  int rxInterruptsCopy = rxInterrupts;
  rxInterrupts = 0;
  interrupts();

  if(printOnce==0 && bufSigned[0]!=0){
    ipPrint();
    printOnce=0;
  }


  //TXbuf[0]++;
  //if(TXbuf[0]==128)
  //  {TXbuf[0] = 0;}

  // Using all the text output to the Serial port is part of the stress test.
  // That causes delays and interrupts.
  if( rxInterruptsCopy > 0)
  {
    Serial.print("Receive: ");
    /*
    if( rxInterruptsCopy > 1) //BY TIME LOOP CAME AROUND,MORE THAN 1 RCV IRPT-
MISSED THOSE
    {
      // Printing to the serial port at 9600 is slow.
      // Therefor it is normal that this sketch misses received data,
      // if too much data was received.
      // As long as the i2c data is correct, everything is okay. It is a stress
test.
      Serial.print("Missed:");
      Serial.print( rxInterruptsCopy);
      Serial.print(" ");
    }
    */
    Serial.print("howMany:");
    Serial.print( rxHowMany);

    Serial.print(", data:");
    for(int i=0; i<rxHowMany; i++)
    {
      if( i == 0)
        Serial.print(F("*"));      // indicate the first number (sometimes used
for a counter value).
```

```
      Serial.print((unsigned int) buffer[i], DEC);
      Serial.print(" ");
    }
    Serial.println();
  }


    convertToSigned();

    Serial.print("BUFSIGNED = ");
    for(int i=0; i<rxHowMany; i++)
    {
      //if( i == 0)
      //  Serial.print(F("*"));      // indicate the first number (sometimes used
for a counter value).

      Serial.print(bufSigned[i], DEC);
      Serial.print(" ");
    }
    Serial.println();

    Serial.print("TXBUF = ");
    for( int m=0; m<10; m++)
    {
      if( m == 0)
        Serial.print(F("*"));      // indicate the number of the counter
      Serial.print( (int) TXbuf[m]);
      Serial.print(F(", "));
    }
    Serial.println();


  //bufInt[0] = (char)buffer[0].toInt();
  Serial.print("BUFFER = ");
  Serial.println(buffer[0]+1);

  noInterrupts();
  boolean flagRequestCopy = flagRequest;
  flagRequest = false;
  interrupts();

  if( flagRequestCopy)
  {
    Serial.println("Request: Data was requested and send");
  }

  // Stress the master by disabling interrupts.
  // A value of 500 microseconds will even corrupt the transmission with the normal
Arduino Wire library.
  //noInterrupts();            //ARS 3 LINES DONT STRESS
  //delayMicroseconds(50);     //ARS DONT STRESS
  //interrupts();              //ARS DONT STRESS
}
```

```
void convertToSigned(void){
  for(int t=0; t<4; t++){      //first 4 elem of buffer are pos ip nbrs
     bufSigned[t] = buffer[t];
    }

  for(int t=4; t<39; t++){   //first 4 elem of buffer are pos ip nbrs
     if(buffer[t]>128){
         bufSigned[t] = -256 + buffer[t];
      }
      else
         bufSigned[t] = buffer[t];

    }
}
```

ArduinoRobotProMini.ino CODE

You can download the code at the following link:

- https://github.com/arshacker/ESP32-ROBOT-CAM-VIDEO-XBOX-CONTROLLER-OPENCV-SYSTEM/blob/main/ArduinoRobotProMini_ANN/ArduinoRobotProMini_ANN.ino

```
// This is the sketch for the Slave Arduino using the hardware I2C.
//ANN 1
#include <Arduino.h>   //oled
#include <U8g2lib.h>   //oled

#include <Wire.h>

#include <SPI.h>   //spi oled

volatile byte buffer[40];
volatile int rxHowMany;
volatile int rxInterrupts = 0;
volatile boolean flagRequest;

signed short int bufSigned[40];
byte TXbuf[] = { 101, 20, 110, 5, 100, 105, 44, 2, 120, 72 }; //ars

int bufInt[40];
int printOnce = 0;

//char basciibuf[20];

U8G2_SH1106_128X64_NONAME_F_4W_HW_SPI u8g2(U8G2_R0, /* cs=*/ 10, /* dc=*/ 9, /*
reset=*/ 8);  //clk pin 13 UNO mosi pin 11 UNO
```

```
//ANN OLED
void ipPrint(void){
  int a = bufSigned[0];
  char asciibuf[20]={0,};
  int b = bufSigned[1];
  int c = bufSigned[2];
  int d = bufSigned[3];

  //char bsciibuf[3];
  itoa(a,asciibuf,10); //10 is decimal
  itoa(b,asciibuf+4,10);
  itoa(c,asciibuf+8,10);
  itoa(d,asciibuf+12,10);

//  itoa(c,basciibuf,10);

  //asciibuf[4]=" ";


  char totalbuf[20];
  totalbuf[0]=asciibuf[0];
  totalbuf[1]=asciibuf[1];
  totalbuf[2]=asciibuf[2];
  totalbuf[3]=asciibuf[4];
  totalbuf[4]=asciibuf[5];
  totalbuf[5]=asciibuf[6];
  totalbuf[6]=asciibuf[7];

  //String myStr1 = String(bufSigned[0]);

  //ug82.drawStr(xcoord=5 x nbr spaces,ycoord=10 x nbr lines,"STRING")

  u8g2.clearBuffer();          // clear the internal memory
  u8g2.setFont(u8g2_font_ncenB08_tr); // choose a suitable font
  u8g2.drawStr(0,10,"IP ADDRESS");  // write 1ST LINE to the internal memory
  u8g2.drawStr(0,20,asciibuf);  // write 2ND LINE to the internal memory
  //u8g2.drawStr(0,20,myStr1);  // write 2ND LINE to the internal memory
  u8g2.drawStr(25,20,asciibuf+4);  // write 2ND LINE to the internal memory
  u8g2.drawStr(50,20,asciibuf+8);  // write 2ND LINE to the internal memory//NEW
  u8g2.drawStr(75,20,asciibuf+12);    //   write   2ND   LINE   to   the   internal
memory//NEW
  //u8g2.drawStr(50,20,asciibuf+4);  // EVEN THIS DOESNT WORK???
  //u8g2.drawStr(50,20,asciibuf+8);  // write 2ND LINE to the internal memory
  //u8g2.drawStr(50,20,"K");
  //u8g2.drawStr(45,20,basciibuf);  // write 2ND LINE to the internal memory
  //u8g2.drawStr(0,30,asciibuf+4);  // write 2ND LINE to the internal memory
  //u8g2.drawStr(0,40,asciibuf+8);  // write 2ND LINE to the internal memory
  //u8g2.drawStr(25,40,asciibuf+12);  // write 2ND LINE to the internal memory
  //u8g2.drawStr(0,50,asciibuf+12);  // write 2ND LINE to the internal memory
  u8g2.drawStr(0,30,"SIGN IN NOW!");  // write LAST LINE to the internal memory
  u8g2.sendBuffer();          // transfer internal memory to the display
  delay(1000);
}
```

```
void testPrint(void){
  u8g2.clearBuffer();          // clear the internal memory
  u8g2.setFont(u8g2_font_ncenB08_tr); // choose a suitable font
  u8g2.drawStr(0,10,"Hello World!");  // write 1ST LINE to the internal memory
  u8g2.drawStr(0,20,"Hello Again World!");  // write 2ND LINE to the internal
memory
  u8g2.drawStr(0,60,"Bye For Now World!");  // write LAST LINE to the internal
memory
  u8g2.sendBuffer();          // transfer internal memory to the display
  delay(1000);
}

//ANN 2
void requestEvent()
{
  static byte x = 0;

  // Fill array with numbers.



  //TXbuf[0] = x++;         // overwrite the first with a counter.
  Wire.write(TXbuf, sizeof(TXbuf));

  flagRequest = true;
}


//ANN 3
void receiveEvent(int howMany)   //HOWMANY BYTES SENT BY MASTER--5 BYTES //ars
{
  for( int i=0; i<howMany; i++)
    buffer[i] = Wire.read();

  rxHowMany = howMany;
  rxInterrupts++;
}

void setup()
{
  Serial.begin(9600);           // start serial for output
  Serial.println("\nSlave 2");

  Wire.begin(6);                // join i2c bus as 2ND slave with address #6
  Wire.onReceive(receiveEvent); // interrupt handler for receiving i2c data
  Wire.onRequest(requestEvent); // interrupt handler for when data is requested by
i2c
  for(int y=0; y<40; y++){
    bufSigned[y] = 0;
  }
  u8g2.begin();   //oled
  delay(10);
  //testPrint();
```

```
}

//ANN 4
void loop()
{
  delay(2000);
  noInterrupts();
  int rxInterruptsCopy = rxInterrupts;
  rxInterrupts = 0;
  interrupts();

  if(printOnce==0 && bufSigned[0]!=0){
    ipPrint();
    printOnce=0;
  }


  //TXbuf[0]++;
  //if(TXbuf[0]==128)
  //  {TXbuf[0] = 0;}

  // Using all the text output to the Serial port is part of the stress test.
  // That causes delays and interrupts.
  if( rxInterruptsCopy > 0)
  {
    Serial.print("Receive: ");
    /*
    if( rxInterruptsCopy > 1) //BY TIME LOOP CAME AROUND,MORE THAN 1 RCV IRPT-
MISSED THOSE
    {
      // Printing to the serial port at 9600 is slow.
      // Therefor it is normal that this sketch misses received data,
      // if too much data was received.
      // As long as the i2c data is correct, everything is okay. It is a stress
test.
      Serial.print("Missed:");
      Serial.print( rxInterruptsCopy);
      Serial.print(" ");
    }
    */
    Serial.print("howMany:");
    Serial.print( rxHowMany);

    Serial.print(", data:");
    for(int i=0; i<rxHowMany; i++)
    {
      if( i == 0)
        Serial.print(F("*"));      // indicate the first number (sometimes used
for a counter value).

      Serial.print((unsigned int) buffer[i], DEC);
      Serial.print(" ");
    }
    Serial.println();
```

```
  }


    convertToSigned();

    Serial.print("BUFSIGNED = ");
    for(int i=0; i<rxHowMany; i++)
    {
      //if( i == 0)
      //  Serial.print(F("*"));      // indicate the first number (sometimes used
for a counter value).

      Serial.print(bufSigned[i], DEC);
      Serial.print(" ");
    }
    Serial.println();

    Serial.print("TXBUF = ");
    for( int m=0; m<10; m++)
    {
      if( m == 0)
        Serial.print(F("*"));      // indicate the number of the counter
      Serial.print( (int) TXbuf[m]);
      Serial.print(F(", "));
    }
    Serial.println();


  //bufInt[0] = (char)buffer[0].toInt();
  Serial.print("BUFFER = ");
  Serial.println(buffer[0]+1);

  noInterrupts();
  boolean flagRequestCopy = flagRequest;
  flagRequest = false;
  interrupts();

  if( flagRequestCopy)
  {
    Serial.println("Request: Data was requested and send");
  }

  // Stress the master by disabling interrupts.
  // A value of 500 microseconds will even corrupt the transmission with the normal
Arduino Wire library.
  //noInterrupts();            //ARS 3 LINES DONT STRESS
  //delayMicroseconds(50);     //ARS DONT STRESS
  //interrupts();              //ARS DONT STRESS
}



void convertToSigned(void){
```

```
  for(int t=0; t<4; t++){      //first 4 elem of buffer are pos ip nbrs
     bufSigned[t] = buffer[t];
    }

  for(int t=4; t<39; t++){   //first 4 elem of buffer are pos ip nbrs
     if(buffer[t]>128){
         bufSigned[t] = -256 + buffer[t];
      }
      else
         bufSigned[t] = buffer[t];

    }
}
```

APPENDIX

LIVING WITH CHROME SECURITY

The work described in this paper used the Chrome browser and Windows 10. The browser client can access OpenCV.js via Internet as used principally in this paper or locally (see commented out section in index.h program above). LOCAL USAGE IS REQUIRED WHEN EMPLOYING SOFTAP.

For local usage, Chrome browser security forbids access to any local file (such as OpenCV.js) in the computer directory unless it is in a local server. In this case, the HTML accesses the local file via the server's URL, and potential security breaches are limited to the server itself.

To create this server, install node.js from https://nodejs.org/en/. Next, create a directory (a convenient place is your root directory and named *server*). Next, place OpenCV.js and any other file your HTML file will need to access, in the *server* directory. Next, open a terminal and type as follows:

```
npm install http-server -g
```

It installs globally, so the command can be typed anywhere. Next, go into the *server* directory and type:

```
http-server
```

The command returns three URLs. Any of these can be used by the HTML client to access any file, including OpenCV, in the *server* directory.

The server can be turned off by typing

```
CTRL C
```

And turned back on by typing as done initially.

```
http-server
```

As might be expected, the SERVER and its URL can also be used if the HTML is run locally.

Remember to place both the JPG file and OpenCV.js in the Server directory before issuing the `http-server` command.

“THE ZERO HACK”

The author was surprised to observe that the ON_EVENT routine in the ESP32 *AsyncWebServer* is not able to transmit a NULL (zero) back to the XBOX Controller program. The presence of a zero in the data caused an error in the XBOX py program.

To overcome this problem the following hack, named “The Zero Hack” has been successfully used.

Consider the following tutorial example of a 7 integer (each integer no greater than one byte large) array containing two zeros at arbitrary locations.

7	6	5	4	3	2	1	0
256,	0,	128,	1,	156,	35,	0,	X

A binary array is formed containing a 1 in each position containing a zero, and a zero in every other position.

7	6	5	4	3	2	1	0
0	1	0	0	0	0	1	X

The X is replaced by the decimal equivalent of the binary number in the above binary array, namely code number 65.

In the original array, the zeros are replaced by ones and the X is replaced by code number 65.

7	6	5	4	3	2	1	0
256,	1,	128,	1,	156,	35,	1,	65

This new array contains no zeros and is successfully transmitted to the XBOX Controller program. The Controller program performs a decimal to binary conversion on code number 65 and recovers the binary number.

7	6	5	4	3	2	1	0
0	1	0	0	0	0	1	65

In each array position corresponding to a binary 1 (position 1 and position 6) the array number in that position is replaced by a zero.

The original array, shown below, is recovered.

7	6	5	4	3	2	1	0
256,	0,	128,	1,	156,	35,	0,	65

CONCLUSIONS

None of the elements of the project described in this paper are new. The ESP32 Camera Web Server, OpenCV, OpenCV.js, and *ESPAsyncWebserver* have each been described extensively and in detail in the literature and referenced here. The novelty here has been the combining of these technologies and including a new XBOXController GUI for robot and OpenCV control.

The ESP32 Camera, with its small size, wi-fi, high tech, and low-cost capability promises to be an interesting front-end capability for robotic applications.

ACKNOWLEDGEMENTS

The author wishes to acknowledge the works of Random Nerd Tutorials which introduced him to ESP8266, ESP32, and ESP32 CAMERA during the Spring, Summer, and Fall of 2020. Rui Santos is an exceptional teacher who has created lucid tutorials in these and related subjects, some free and others are offered at modest prices. He is surprisingly accessible and the RNT Facebook group is very helpful.

I would also be extremely remiss should I not acknowledge the very valuable and inciteful collaboration with Sara Santos whose advice and active participation have turned much of this work into understandable prose for readers. Without her involvement, this paper would not have seen the light of day.

This work was started and continued during the pandemic of 2020 and played a significant part in focusing the author away from its terrible events. The author is also indebted to his “pande-mate”, loyal wife, and bridge partner Gerry with whom he sheltered for more than a year.

AUTHOR BACKGROUND

Andrew (“DOC”) R. Sass holds a BSEE(MIT), MSEE & PhD EE (PURDUE). He is a retired research engineer (integrated circuit components), a second-career retired teacher (AP Physics, Physics, Robotics), and has been a mentor of a local FIRST robotics team.