LUKE ARM - HUMAN EMULATING ROBOTIC HAND

Project report

EKLAVYA MENTORSHIP PROGRAM AT SOCIETY OF ROBOTICS AND AUTOMATION,
VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE ,
MUMBAI.

SEPTEMBER-OCTOBER 2022.

ACKNOWLEDGEMENT

WE ARE TRULY GRATEFUL FOR THIS WONDERFUL OPPORTUNITY THAT SRA HAS BESTOWED ON US. THANKYOU MENTORS AND SENIORS FOR YOUR GUIDANCE, SUPPORT, MOTIVATION. IT WAS IMPOSSIBLE TO ACHIEVE THIS WITHOUT YOU ALL. WE'LL LOVE TO WORK AND ACHIEVE FUTURE GOALS TOGETHER.

SPECIAL THANKS TO ;

OM SHELADIA
RISHIKESH DONADKAR

TEAM MEMBERS:

KAMAKSHI DHOKEY
+8879640952
k.dhokey@gmail.com

VAIDIC GUPTA
++918857826380

1] PROJECT OVERVIEW :

In this project we will establish connection between two ESP32 boards (mainly focusing on the ESP-NOW protocol). One board will be the master and the other the slave, controlling servos as per command.

2] TECHNOLOGY USED:

1)ESP-NOW, is a protocol developed by Espressif. It is a fast communication protocol that can be used to exchange small messages (up to 250 bytes) between esp32 boards.
The protocol is similar to the low-power 2.4GHz wireless connectivity that is often deployed in wireless mouses. So, the pairing between devices is needed prior to their communication.
After the pairing is done, the connection is secure and peer-to-peer, with no handshake being required.
ESP-NOW is versatile and you can have one-way or two-way communication
between two esp32 boards.

Why use ESP-NOW?

● It allows ESP devices to communicate directly without connecting to a WiFi
network

● The pairing between devices is needed prior to their communication. After the pairing is done, the connection is secure and peer-to-peer. If suddenly one of your boards loses power or resets, when it restarts, it will automatically connect to its peer to continue the communication.

It does not require a router to connect two esp32 boards.Thus, it can be used at any remote places.

● It provides encrypted and unencrypted unicast communication.

● It does not need to connect to a Wifi access point which makes it a fast communication protocol .

● Maximum 20 nodes can be connected.

3] PROJECT IDEA:

Aim of the project is designing a cost effective arm housing 5 servos controlled by one slave esp-32 taking commands from a remote esp-32 by establishing wifi communication between the two,essentially one acting as a server and the other as a client.

The user will have to relay finger flex data to the arm and control the servos using client esp accordingly.

4] BASIC PROJECT DOMAINS:

Embedded C

5] WORKFLOW :

1)ESP-NOW protocol:

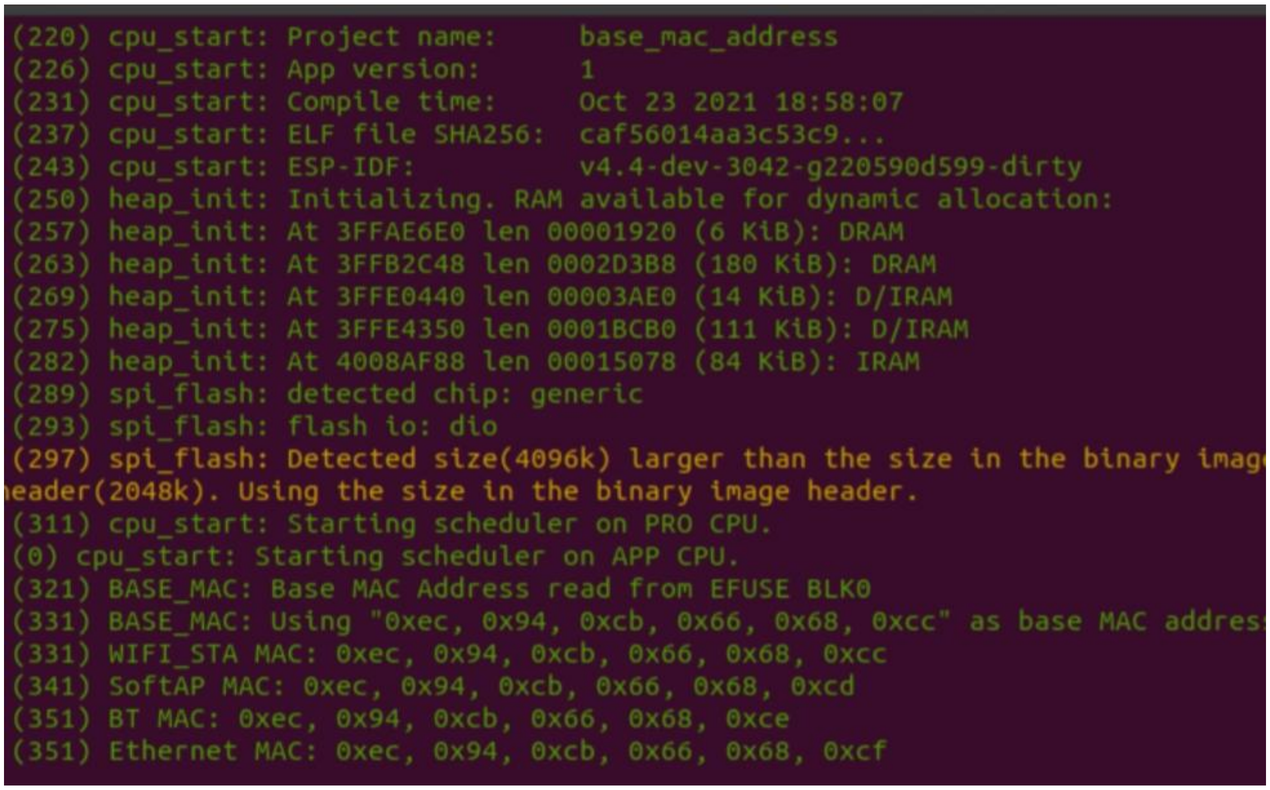
https://github.com/espressif/esp-idf/tree/master/examples/wifi/espnow

While using the above espnow-example to send data, you will come across a lot
of ‘extras’ on top of espnow data.
These ‘extras’ are not required to send the use the ESPNOW protocol, however to make the data more safe and reliable it is recommended that we use them.

We will require two esp devices for this project. While sending data these two
devices must be on the same channel, must have the same primary and local
master key.
The maximum data that can be transferred is of 250 bytes which can be changed under “Example Configuration”.
In our project we will be using the code snippets form the following ESPNOW
code.

https://github.com/Mair/esp32-course/tree/master/_18_ESP_NOW/_18_esp_now

It is not necessary to know the mac address of the esp. To find out the mac



```
(220) cpu_start: Project name:     base_mac_address
(226) cpu_start: App version:      1
(231) cpu_start: Compile time:     Oct 23 2021 18:58:07
(237) cpu_start: ELF file SHA256:  caf56014aa3c53c9...
(243) cpu_start: ESP-IDF:          v4.4-dev-3042-g220590d599-dirty
(250) heap_init: Initializing. RAM available for dynamic allocation:
(257) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
(263) heap_init: At 3FFB2C48 len 0002D3B8 (180 KiB): DRAM
(269) heap_init: At 3FFE0440 len 00003AE0 (14 KiB): D/IRAM
(275) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
(282) heap_init: At 4008AF88 len 00015078 (84 KiB): IRAM
(289) spi_flash: detected chip: generic
(293) spi_flash: flash io: dio
(297) spi_flash: Detected size(4096k) larger than the size in the binary imag
eader(2048k). Using the size in the binary image header.
(311) cpu_start: Starting scheduler on PRO CPU.
(0) cpu_start: Starting scheduler on APP CPU.
(321) BASE_MAC: Base MAC Address read from EFUSE BLK0
(331) BASE_MAC: Using "0xec, 0x94, 0xcb, 0x66, 0x68, 0xcc" as base MAC addres
(331) WIFI_STA MAC: 0xec, 0x94, 0xcb, 0x66, 0x68, 0xcc
(341) SoftAP MAC: 0xec, 0x94, 0xcb, 0x66, 0x68, 0xcd
(351) BT MAC: 0xec, 0x94, 0xcb, 0x66, 0x68, 0xce
(351) Ethernet MAC: 0xec, 0x94, 0xcb, 0x66, 0x68, 0xcf
```

6] CHALLENGES FACED

- We were unable to use more flex sensors due to unavailability of velostat paper, and high cost of flex sensors
- We didn't get accurate readings from flex sensors made up of pencil shade.

7] FUTURE WORK

- use four more flex sensors, each controlling one servo motor.

8] REFERENCE LINKS:

1) Embedded C :

freeRTOS Playlist:

https://youtube.com/playlist?list=PLXyB2ILBXW5FLc7j2hLcX6sAGbmH0JxX8

freeRTOS study-group
https://sravjti.in/embedded-systems-study-group/week6/week6.html

API reference:
https://www.freertos.org/a00106.html

API reference for espnow:

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html

Esp-idf project build system-

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/build-system.html

2) Designing

   https://youtu.be/QOyghUxLdqE

   https://youtu.be/5z5evInThP4

3) Servo motor control

https://github.com/espressif/esp-idf/blob/master/examples/peripherals/mcpwm/mcpwm_servo_control/main/mcpwm_servo_control_example_main.c

4) ADC flex sensor testing

https://esp32tutorials.com/esp32-adc-esp-idf/