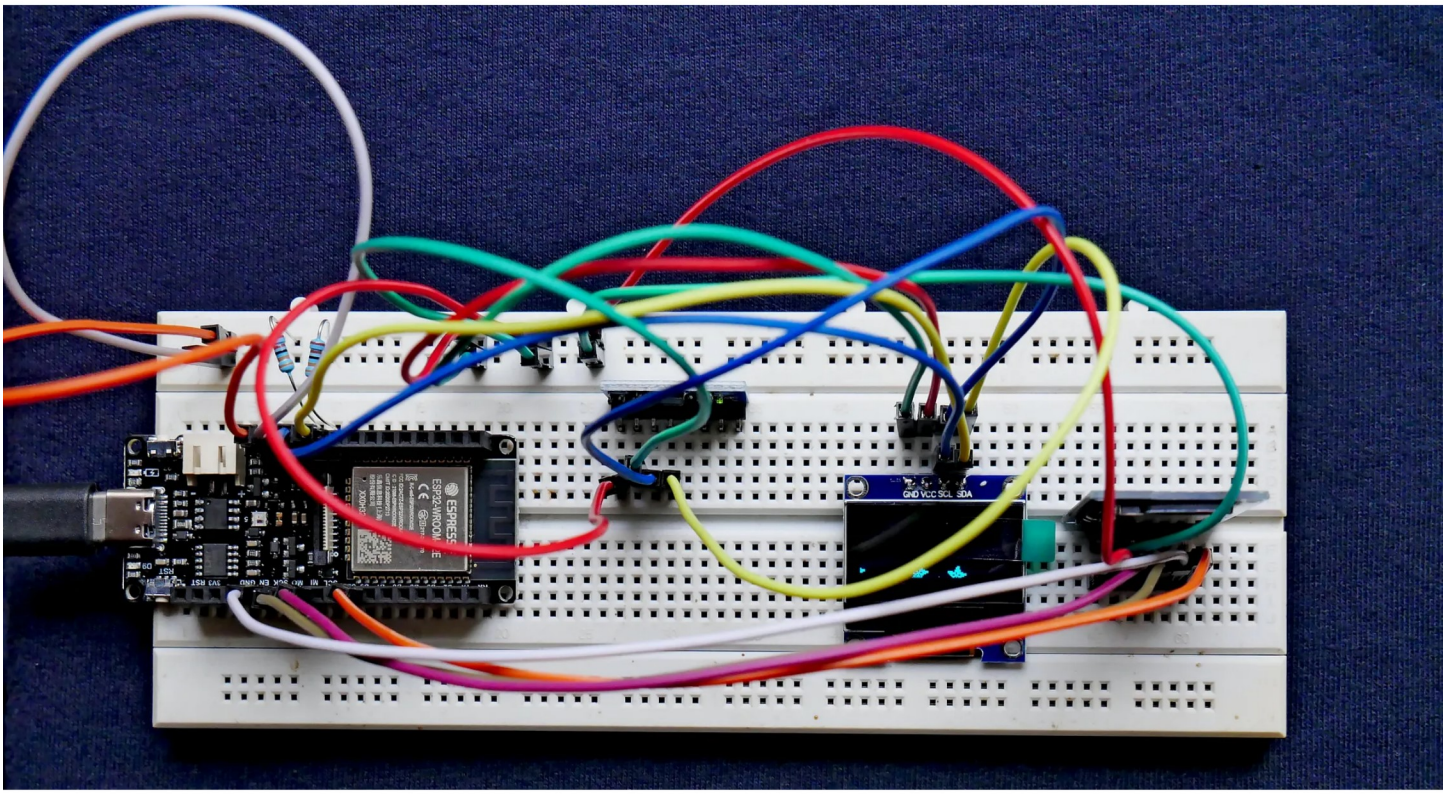


ESP32 Firmware Encryption

Kattelroshan · Follow
5 min read · May 25

The default MEMORY offsets in PlatformIO:

1. Bootloader: **0x1000**

2. Partition Table: **0x8000**

3. Firmware: **0x10000**

We can read the firmware file inside the esp32 kit by using the following command

```
esptool.py --port PORT --baud 230400 read_flash 0 0x40000 ./trick.bin
```

The firmware file is stored as trick.bin and you can read this file using any of the HEX reader plugins in VScode.

Steps for Encryption

1. Clone the Esptool GitHub folder and execute the following commands

```
$ python setup.py install                  // To install from setup.py file
$ espefuse.py --port "COM12" Summary        //press the boot button now
```

> *Here it will show the map of the chip's internal fuses. Mainly we can focus on the following parameters.*

Inside Security Fuses :

FLASH_CRYPT_CNT =0 , FLASH_CRYPT_CONFIG = 0

- The flash encryption key will be stored in

BLOCK1 (BLOCK1): Flash encryption key

= 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 R/W

- The secure boot encryption key will be stored in

BLOCK2 (BLOCK2): Secure boot key

= 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 R/W

2. Compile the source code and generate the "firmware.bin" file using any platform such as Arduino UNO, or PlatformIO.

3. Generate flash encryption Key for the esp32

```
$ espsecure.py generate_flash_encryption_key
"<path_to_bin_file_to_store_key>"

eg:
$ espsecure.py generate_flash_encryption_key flash_encryption_key.bin
```

> *This process is irreversible only do this when the project is completely set up (In production)*

4. Burn the key on your esp32

> *Press the boot button and Type BURN*

```
$ espefuse.py --port PORT burn_key flash_encrytion
"<path_to_bin_file_to_store_key>"

eg:
$ espefuse.py --port COM12 burn_key flash_encryption
flash_encryption_key.bin
```

5. Now we can again read the summary and see the flash encryption key

```
$ espefuse.py --port "COM12" Summary
```

Now block 1 will look something like this

BLOCK1 (BLOCK1): Flash encryption key

= ?? -/-

6. Enabling flash encryption mechanism

> *Press the boot button and Type BURN*

```
$ espefuse.py --port COM12 burn_efuse FLASH_CRYPT_CNT
```

7. Configuring Flash encryption to use all address bits together with Encryption Key (max value 0×0F)

> *Press the boot button and Type BURN*

```
$ espefuse.py --port COM12 burn_efuse FLASH_CRYPT_CONFIG 0x0F
```

8. Now again check the summary of the espefuse

```
$ espefuse.py --port "COM12" Summary
```

Now you can see inside Security Fuses -

FLASH_CRYPT_CNT =1 , FLASH_CRYPT_CONFIG = 15

9. Generate encryption for the partition, bootloader and firmware

- **Encrypting the partition table**

```
$ espsecure.py encrypt_flash_data --keyfile "<path to key.bin>key.bin"
--address 0x8000 --output path to store encrypted partition table.bin>"
"<path to unencrypted partition.bin>"

eg:
$ espsecure.py encrypt_flash_data --keyfile flash_encryption_key.bin
--address 0x8000 --output partition_table_encrypted.bin
build/partition_table/partition-table.bin
```

This command encrypts the partition table binary file (**`partition-table.bin`**) using

the flash encryption key and outputs the encrypted partition table to **`partition_table_encrypted.bin`** at the address 0x8000.

- **Encrypting the bootloader**

```
$ espsecure.py encrypt_flash_data --keyfile "<path to key.bin>key.bin" -
address 0x1000 --output "<path to store encrypted bootloader.bin>" "<path
to unencrypted bootloader.bin>"

eg:
$ espsecure.py encrypt_flash_data --keyfile flash_encryption_key.bin
--address 0x1000 --output bootloader_encrypted.bin build/bootloader
/bootloader.bin
```

This command encrypts the bootloader binary file (**`bootloader.bin`**) using the flash encryption key stored in **`flash_encryption_key.bin`** . The encrypted bootloader is then outputted to **`bootloader_encrypted.bin`** at the address 0x1000.

- **Encrypting the firmware:**

```
$ espsecure.py encrypt_flash_data --keyfile "<path to key.bin>key.bin"
--address 0x10000 --output "<path to store encrypted firmware.bin >"
"<path to unencrypted firmware.bin>"

eg:
$ espsecure.py encrypt_flash_data --keyfile flash_encryption_key.bin
--address 0x10000 --output firmware_encrypted.bin firmware.bin
```

- This command encrypts the firmware binary file (**`firmware.bin`**) using the flash encryption key and outputs the encrypted firmware to **`firmware_encrypted.bin`** at the address 0x10000.

These commands demonstrate how to use **`espsecure.py`** to encrypt specific sections of the firmware using the provided flash encryption key. The resulting encrypted

files can then be flashed onto the ESP32 device.

> *Make sure you have the necessary encryption key (**`flash_encryption_key.bin`**) and the respective binary files (**`bootloader.bin`** , **`partition-table.bin`** , and **`firmware.bin`**) in the specified locations before running these commands.*

10. Finally burn these files to the esp32 board using the following command and press the boot button

```
$ python esptool.py --chip esp32 --port COM8 --baud 460800 --before
default_reset --after hard_reset write_flash -z --flash_mode dio
--flash_freq 40m --flash_size detect 0x1000 "<path to encrypted
bootloader.bin file>" 0x10000 "<path to encrypted firmware.bin file >"
0x8000 "<path to encrypted partition.bin file>"

eg:
$ esptool.py -p PORT -b 460800 --before default_reset --after hard_reset
--chip esp32  write_flash --flash_mode dio --flash_size detect
--flash_freq 40m 0x1000 bootloader_encrypted.bin 0x8000
partition_table_encrypted.bin  0x10000 firmware_encrypted.bin
```

11. Check whether the encrypted mechanism added is working or not using the following Arduino code

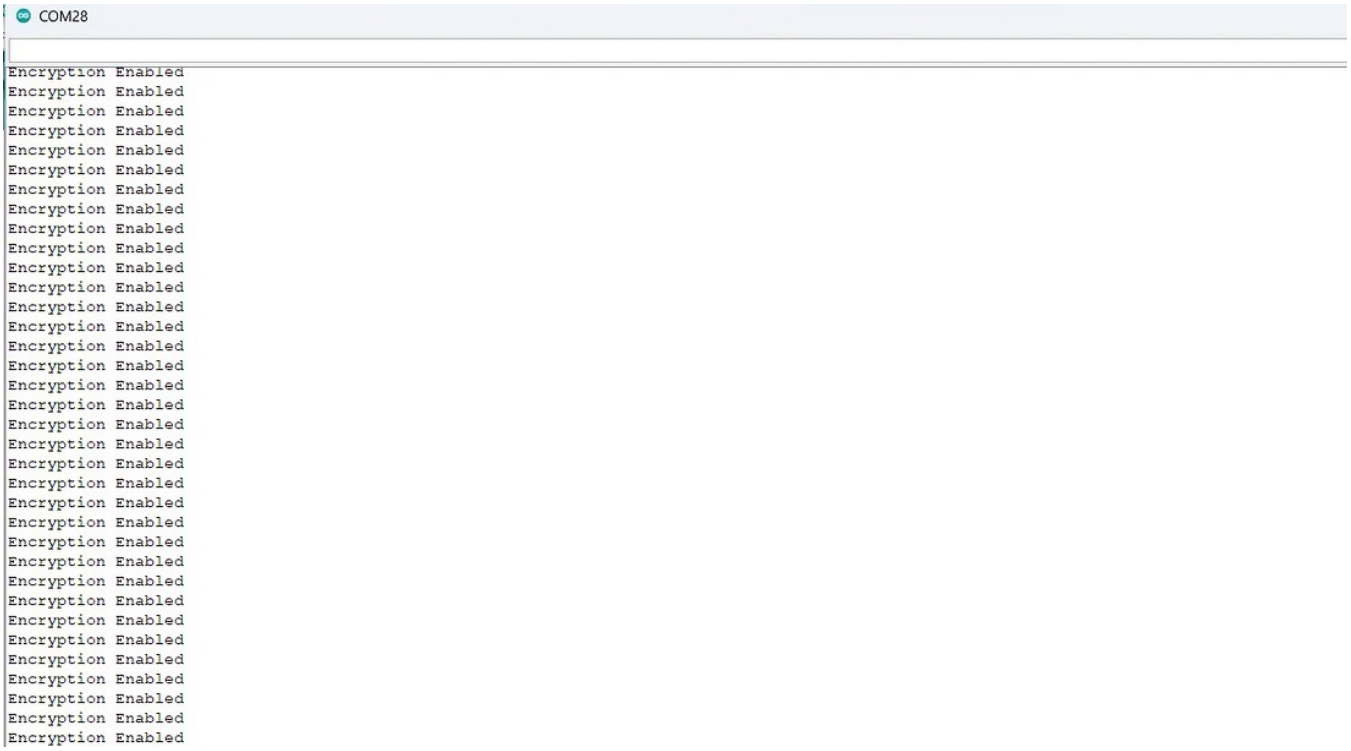
> *Generate the firmware file and make sure it is encoded with the previously generated key*

```
# include "esp_flash_encrypt.h"
void setup() {
  Serial.begin(115200);
}

void loop() {
  if (esp_flash_encryption_enabled())
    Serial.println("Encryption Enabled");
  else
    Serial.println("Encryption not Enabled");
}
```

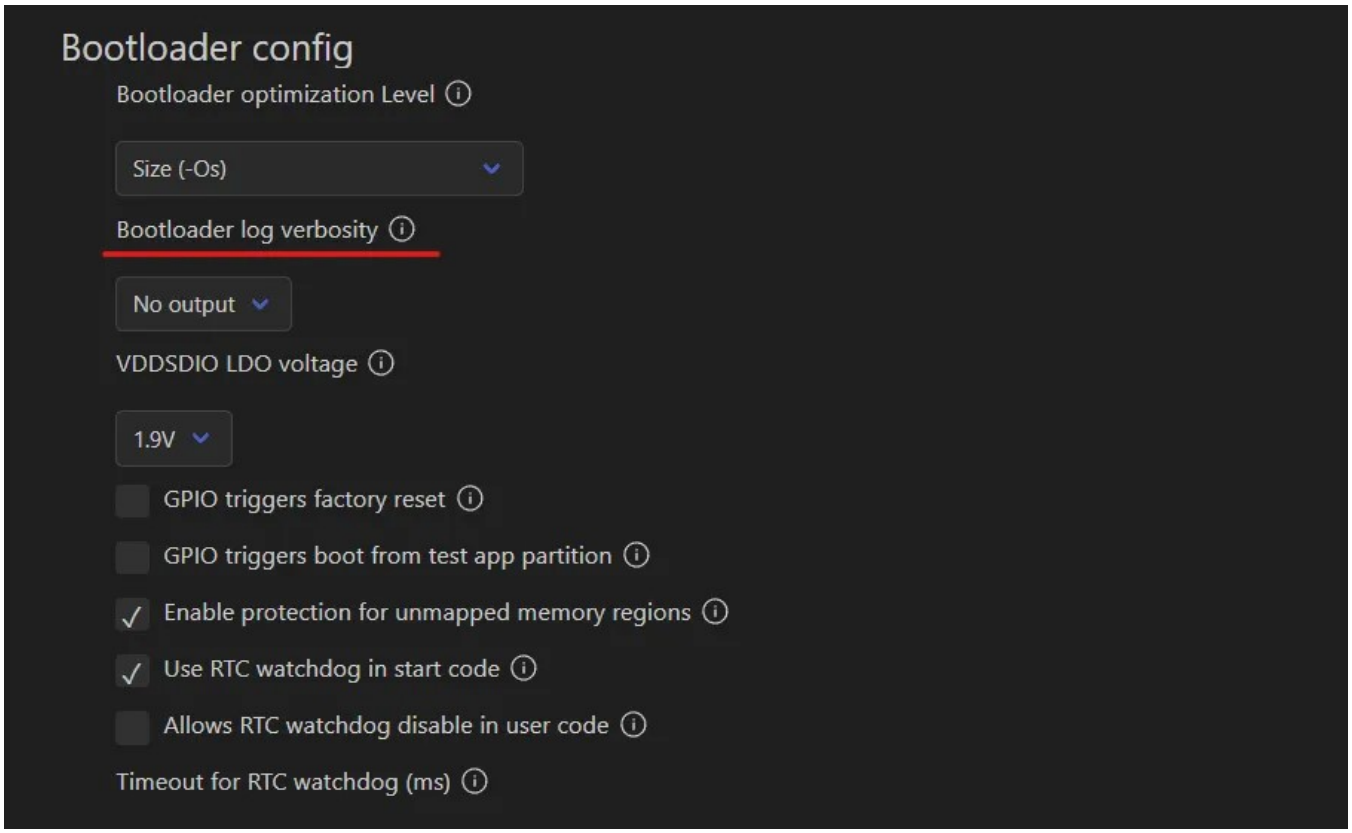
> *→In order to Reprogram the board we have to generate a new firmware file and encrypt it with the existing KEY.bin file (flash_encryption_key.bin) then only upload the code to the model*
>
> *→We Have to generate new flash_encryption_key.bin file for each esp32 module*

After you upload the encrypted test code, if you get this output in the serial terminal then Congratulation 👏👏👏 you have successfully enabled the encryption mechanism in your esp32 board.

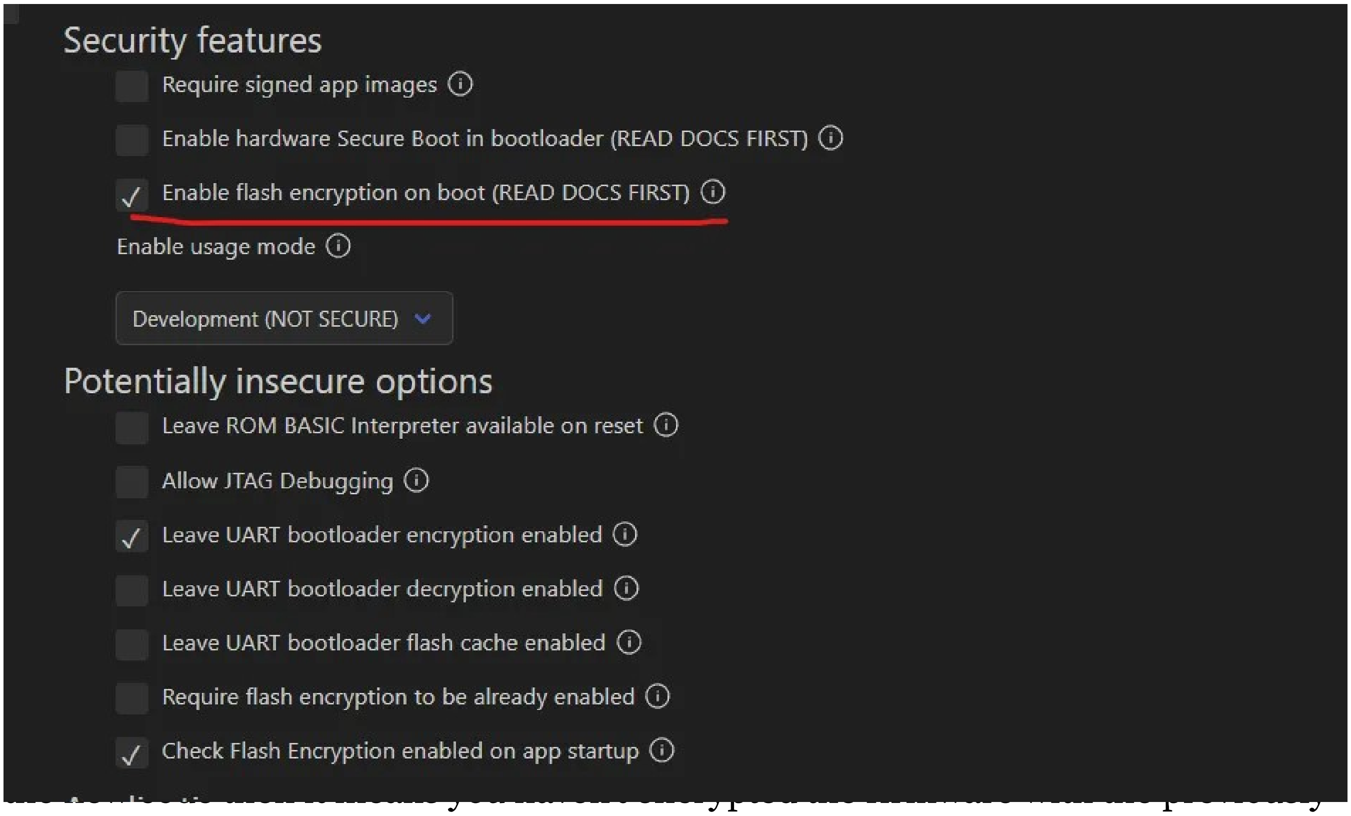


We can use ESP-IDF to generate the bootloader.bin file. Follow these steps to install the IDF. Use the Hello_world program to generate the bootloader.bin file and set the following parameters in menu-config.

1. **Set the Bootloader log verbosity to (no output) — this will decrease the size of the bootloader, so you can enable flash encryption without changing the partition-table offset in PlatformIO and in the IDF.**



2. Enable Flash Encryption (for testing the procedure, set the usage mode to develop only)



generated key.

3. Double-check the partition scheme and offset of the partition table to address 0x8000

Error Serial Terminal log

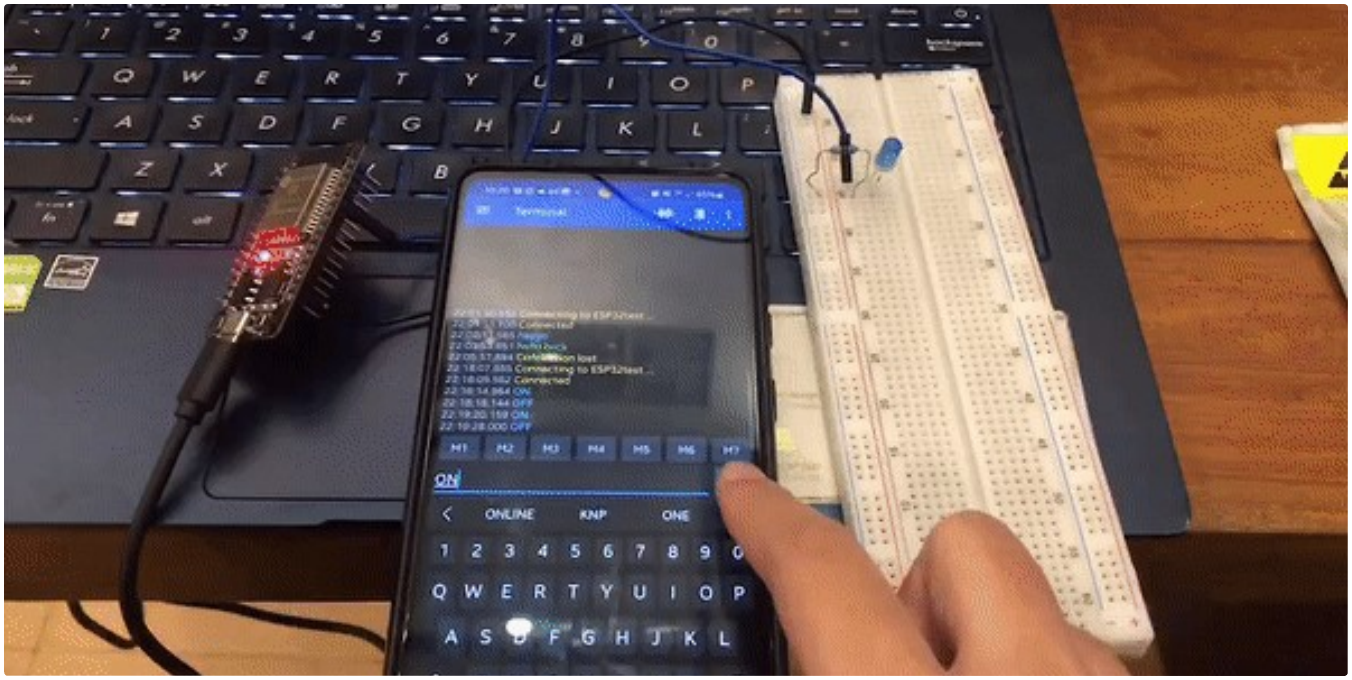
Esp32 Security Encryption IoT Programming

Follow

Written by Kattelroshan

1 Follower

Recommended from Medium



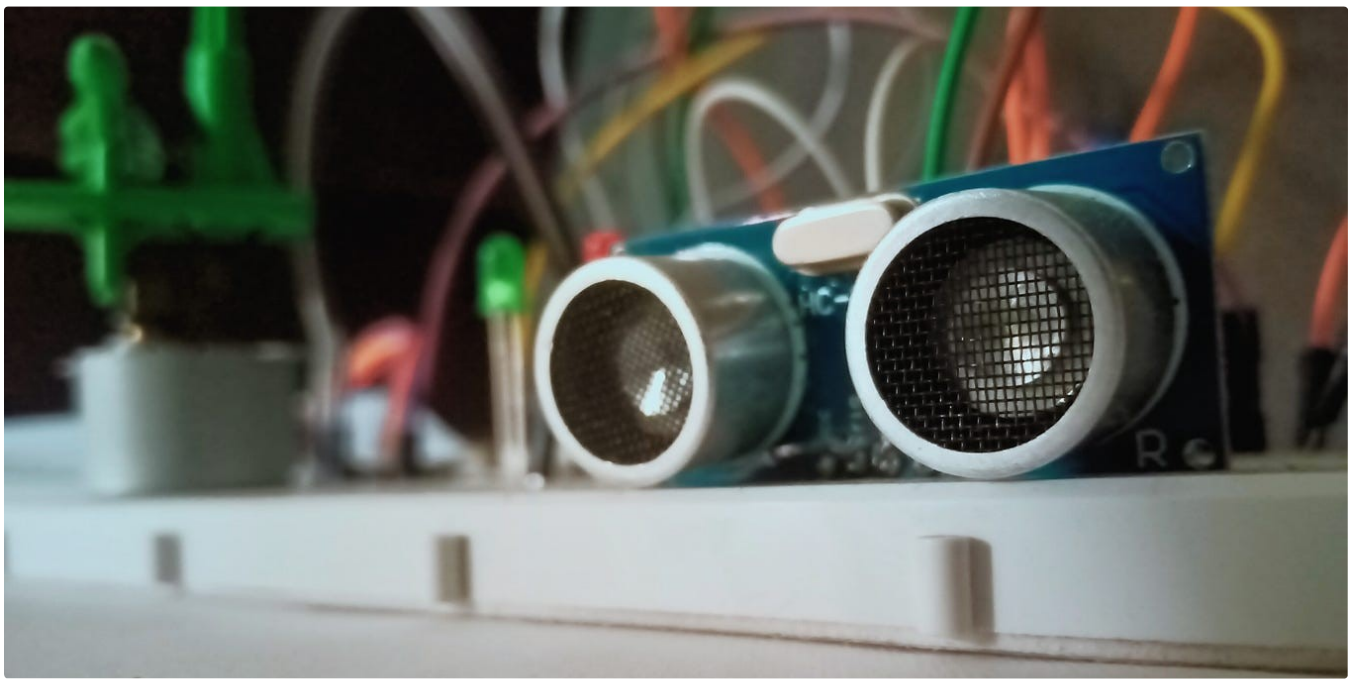
Raniakiranaa

ESP32 Project: Bluetooth

a step-by-step guide

6 min read · Mar 26

16



Shambaditya Mukherjee

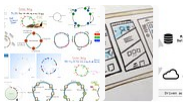
IoT Project with ESP32 Ultrasonic and Stepper Motor

Project goal:-

6 min read · May 25

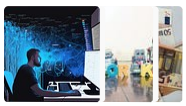
8

Lists



General Coding Knowledge

20 stories · 322 saves

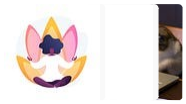


It's never too late or early to start something

15 stories · 113 saves

Coding & Development

11 stories · 164 saves



Stories to Help You Grow as a Software Developer

19 stories · 362 saves

Beoloq

Specialities of technologies of Aleo

Hello my dear friends! My nick is Beoloq! I want to talk abot features, funcrions and technical specialities of Aleo!

3 min read · Jul 26

2

Tejesh Kumar in Towards Dev

Makefile: An Introduction

We all need things to be fast and easy. Faster CPU. Faster Memory. Faster Communication. Then, why is your C/C++ compilation still slow...

4 min read · Aug 13

17 2

rootisgod

5 Essential Tools to Run Your Own Homelab

5 min read · Jul 19

463 2


Sachini Rasanga

Cryptography

Overview of this blog?

3 min read · Aug 16



See more recommendations