# Panic Button v2

Merupakan upgrade dari versi sebelumnya yaitu v1.1. yang paling membedakan adalah controller diupgrade ke esp32 yang sebelumnya esp12-E.

## Hardware

Untuk board, tidak ada perubahan dari segi bentuk dan dimensi, masih sama dengan versi sebelumnya.

A green circuit board with many small chips

AI-generated content may be incorrect.

Gambar Panic Button V1.1

A green circuit board with black and white text

AI-generated content may be incorrect.

Gambar Panic Button V2.1

Perubahan yang dapat diperhatikan adalah sebagai berikut:

* Digunakan esp32, letak antenna di luar pcb.
* Rangkaian IC NE555 dihilangkan.
* Ditambahkan regulator 3V3.
* Ditambahkan FET untuk ADC baterai.
* Ditambahkan driver untuk buzzer.
* Header untuk upload code dipindah ke atas.
* Connector antar kedua pcb menggunakan header siku.

### 3.3V regulator

A diagram of a circuit

AI-generated content may be incorrect.

Gambar LDO 3.3V

Digunakan LDO (low dropout) regulator 3.3V AP2112K, keunggulan regulator tersebut dibanding AMS1117 adalah pada sisi dropoutnya lebih kecil. Sehingga pada saat tegangan baterai sudah ada di 3.7V regulator tersebut masih bisa stabil mengeluarkan tegangan di 3.3V. pada versi 1.1 tidak terdapat regulator sama sekali, esp langsung mendapat tegangan dari baterai.

### Enable untuk cek adc baterai

A diagram of a circuit

AI-generated content may be incorrect.

Gambar rangkain FET untuk adc baterai

Ditambahkan fet untuk enable adc baterai, sehingga pada saat tidak melakukan pengukuran baterai maka potensi power lost pada resistor pembagi tegangan sebelum adc dapat dihilangkan.

### Penambahan komunikasi RS45

A diagram of a circuit board

AI-generated content may be incorrect.

Gambar rangkaian MAX485

Juga ditambahkan untuk komunikasi max485, antisipasi jika ingin digunakan dengan kabel.

## Firmware

Untuk firmware, terdapat 2 jenis tombol dan 2 jenis komunikasi yang dapat digunakan. Perbedaan tersebut dapat dipilih melalui flag yang ada, flag tersebut yang kemudian akan digunakan untuk memilih task mana yang diaktifkan.

ButtonType buttonType = BUTTON\_NC\_LOCKED;

// ButtonType buttonType = BUTTON\_NO\_MOMENTARY;

ConnectivityType connectionType = CONNECTION\_WiFi;

// ConnectivityType connectionType = CONNECTION\_ESPNOW;

void setup() {

  // put your setup code here, to run once:

  pinConfig();

  Serial.begin(115200);

  batteryCalculate();

  if (connectionType == CONNECTION\_WiFi) {

    xTaskCreate(task\_wifi, "wifi", 1024 \* 8, NULL, 3, NULL);

  } else if (connectionType == CONNECTION\_ESPNOW) {

    xTaskCreate(task\_espnow, "esp now", 1024 \* 3, NULL, 2, NULL);

  }

  if (buttonType == BUTTON\_NC\_LOCKED) {

    xTaskCreate(task\_button, "button", 1024 \* 2, NULL, 1, NULL); // locked NC switch

  } else if (buttonType == BUTTON\_NC\_LOCKED) {

    xTaskCreate(task\_button2, "button2", 1024 \* 2, NULL, 1, NULL); // momentary NO switch

  }

  xTaskCreate(task\_led, "led", 1024, NULL, 1, NULL);

  xTaskCreate(task\_buzzer, "buzzer", 1024, NULL, 1, NULL);

}

### Button locked NC

Button locked NC merupakan

void task\_button(void \*pvParameter) {

  esp\_sleep\_enable\_ext0\_wakeup(PIN\_WAKE\_UP, HIGH);

  if (digitalRead(PIN\_WIFI\_RESET) == LOW) {

    if (connectionType == CONNECTION\_WiFi) {

      status.apMode = true;

      Serial.println("apMode detected");

    } else if (connectionType == CONNECTION\_ESPNOW) {

      pairingMode = true;

      Serial.println("Pairing Mode detected");

    }

  } else {

    if (connectionType == CONNECTION\_WiFi) {

      status.apMode = false;

      Serial.println("no apMode detected");

    } else if (connectionType == CONNECTION\_ESPNOW) {

      pairingMode = false;

      Serial.println("no Pairing Mode detected");

    }

  }

  while (1) {

    if (digitalRead(PIN\_BUTTON\_MAIN) == HIGH) {

      status.emergency = true;

      led.offDelay = 100;

    } else if (digitalRead(PIN\_BUTTON\_MAIN == LOW)) {

      status.emergency = false;

      led.offDelay = 500;

    }

    vTaskDelay(10);

  }

}

Jika pada saat awal nyala (baik power on atau wake up). Tombol wifi reset ditekan, maka akan mengaktifkan apMode untuk memasukkan WiFi config atau pairing esp now. Sedangkan pada saat wifi reset tidak tertekan, maka akan emergency seperti biasa.

Jika tombol tertekan maka akan mendeteksi emergency, sedangkan pada saat dilepas status emergency tersebut dihilangkan.

### Button momentary NO

void task\_button2(void \*pvParameter) {

  esp\_sleep\_enable\_ext0\_wakeup(PIN\_WAKE\_UP, LOW);

  Button mainButton(PIN\_BUTTON\_MAIN);

  uint8\_t counter = 0;

  esp\_reset\_reason\_t resetReason = esp\_reset\_reason();

  if (digitalRead(PIN\_WIFI\_RESET) == LOW) {

    status.apMode = true;

    pairingMode = true;

    Serial.println("apMode detected");

  } else {

    status.apMode = false;

    pairingMode = false;

    Serial.println("no AP mode");

  }

  while (digitalRead(PIN\_BUTTON\_MAIN) == LOW) {

    vTaskDelay(100);

    counter++;

    if (counter > 30)

      break;

  }

  if (counter < 30) { // pressed less than 3 second after start

    if (!status.apMode) {

      if (resetReason == ESP\_RST\_DEEPSLEEP) {

        status.emergency = true;

        log\_i("emergency true");

        led.offDelay = 100;

      }

    }

  } else {

    pairingMode = true;

  }

  while (1) {

    if (mainButton.read() == LOW) {

      if (!mainButton.pressed) {

        mainButton.pressed = true;

        mainButton.startTime = millis();

      }

      if (millis() - mainButton.startTime > 1000 && !mainButton.longPressed) {

        mainButton.longPressed = true;

        // button long pressed

        log\_i("main button long pressed");

        if (status.emergency == true) {

          status.emergency = false;

          led.offDelay = 500;

        }

        vTaskDelay(500);

      }

    } else { // button released

      if (mainButton.pressed) {

        if (mainButton.longPressed) {

          mainButton.longPressed = false;

        } else {

          // button short pressed

          log\_i("main button short pressed");

          if (!pairingMode) {

            if (!status.emergency) {

              status.emergency = true;

              log\_i("emergency true");

              led.offDelay = 100;

            }

          }

        }

        vTaskDelay(100);

        mainButton.pressed = false;

      }

    }

    vTaskDelay(10);

    taskYIELD();

  }

}

Karena button adalah momentary switch, pada saat awal device menyala akan dicek terlebih dahulu apakah tombol ditekan selama 3 detik atau kurang. Jika lebih dari 3 detik maka akan masuk pada mode ap untuk set wifi config atau pairing esp-now.

Jika kurang dari itu maka akan menyalakan status emergency dan langsung masuk pada loop. Karena tombol berupa momentary switch, untuk mematikan harus menekan tombol lagi. Untuk pengaman jika tombol dipencet berulang tidak nyala mati terus menerus, maka untuk mematikan digunakan long press selama 1 detik.

### Wifi mqtt

void task\_wifi(void \*pvParameter) {

  iot\_init();

  int counter = 0;

  bool sleep\_ok = false;

vTaskDelay(500);

  while (1) {

    vTaskDelay(10);

    iot\_loop();

    if (WiFi.status() == WL\_CONNECTED || WiFi.status() == WL\_CONNECT\_FAILED) {

      sleep\_ok = true;

    }

    if (iot\_isConnected()) {

      led.onDelay = 100;

      buzzerDelay = 400;

      // send battery information

      if (battery.isCharging) {

        iot\_publish("sensor/status", "Charging");

      } else {

        if (battery.voltage > 4.8) {

          iot\_publish("sensor/status", "No Battery");

        } else {

          iot\_publish("sensor/status", "On Battery");

        }

      }

      // batt

      char batteryPercentageBuffer[3];

      sprintf(batteryPercentageBuffer, "%d", battery.percentage);

      iot\_publish("sensor/battery", batteryPercentageBuffer);

      // send emergency

      while (status.emergency) {

        // timer for publish, 1 count = 1s, publish every 10 second.

        if (counter == 0) {

          iot\_publish("sensor/emergency", "true");

          if (SIRENE\_SN != "") {

            char sirene\_topic[50];

            sprintf(sirene\_topic, "1/%s/relay/onoff/set", SIRENE\_SN);

            iot\_publish(sirene\_topic, "true");

          }

        }

        Serial.println(counter);

        vTaskDelay(1000);

        counter == 10 ? counter = 0 : counter++;

      }

      // if not emergency, and emergency already done.

      iot\_publish("sensor/emergency", "false");

      if (SIRENE\_SN != "") {

        char sirene\_topic[50];

        sprintf(sirene\_topic, "1/%s/relay/onoff/set", SIRENE\_SN);

        iot\_publish(sirene\_topic, "false");

      }

    } else {

      led.onDelay = 500;

    }

     if (!(status.emergency || status.apMode)) {

       if (sleep\_ok) {

         Serial.println("sleeping");

        iot\_publish("$state", "sleeping");

        delay(3000);

        ESP.deepSleep(24 \* HOUR\_MULTIPLIER);

      }

    }

  }

}

Untuk task wifi mqtt, pertama dilakukan init mqtt seperti biasanya. Kemudian pada loop, dilakukan pengecekan apakah wifi sudah terhubung atau gagal, jika terhubung maka akan publish untuk baterai information, jika status.emergency true maka akan publish ke server tiap 10 detik. jika sudah keluar dari status emergency, maka akan publish emergency false dan esp akan sleep.

### Esp-now

void task\_espnow(void \*pvParameter) {

void task\_espnow(void \*pvParameter) {

  bool temp;

  bool emergencyEnd = false;

  my\_espnow\_init();

  while (1) {

    if (pairingMode) {

      my\_espnow\_pairing();

      pairingMode = false;

    }

    // wait for emergency status true, max 3 second

    for (int i = 0; i < 30; i++) {

      if (status.emergecy) {

        break;

      }

      vTaskDelay(100);

    }

    while (status.emergency) {

      myData.msgType = DATA;

      myData.emergencyStatus = status.emergency;

      myData.batteryStatus = battery.isCharging;

      myData.batteryPercentage = uint8\_t(battery.percentage);

      esp\_now\_send(NULL, (uint8\_t \*)&myData, sizeof(myData));

      vTaskDelay(1000);

    }

    if (!status.emergency) {

      myData.msgType = DATA;

      myData.emergencyStatus = status.emergency;

      esp\_now\_send(NULL, (uint8\_t \*)&myData, sizeof(myData));

    }

    if (!(status.emergency || pairingMode)) {

      Serial.println("sleeping");

      vTaskDelay(500);

      ESP.deepSleep(24 \* HOUR\_MULTIPLIER);

    }

  }

  vTaskDelay(1);

}

Untuk task esp now, dilakukan init kemudian masuk ke dalam loop, seperti pada task wifi, jika status.emergency true maka akan kirim data pada gateway melalui esp-now. Jika sudah keluar dari status emergency maka langsung mengirim false dan sleep esp.

void my\_espnow\_init(void) {

  getSerial();

  file.init();

  String fileJson = file.readFile(SPIFFS, "/peer.json");

  log\_i("here");

  deserializeJson(peerDataJson, fileJson);

  serializeJsonPretty(peerDataJson, Serial);

  if (!peerDataJson.isNull()) {

    const char \*temp = peerDataJson["peer\_addr"];

    uint8\_t temp2[6];

    for (int i = 0; i < 6; i++) {

      temp2[i] = (uint8\_t)temp[i];

    }

    memcpy(peerInfo.peer\_addr, temp2, 6);

    Serial.println(peerInfo.peer\_addr[1]);

    peerInfo.channel = peerDataJson["channel"];

    Serial.println(peerInfo.channel);

    peerInfo.encrypt = peerDataJson["encrypt"];

    Serial.println(peerInfo.encrypt);

  }

  WiFi.mode(WIFI\_MODE\_STA);

  getMacAddress();

  if (esp\_now\_init() != ESP\_OK) {

    Serial.println("ESP NOW Init Failed");

  }

  esp\_wifi\_set\_channel(peerInfo.channel, WIFI\_SECOND\_CHAN\_NONE);

  // Add peer

  if (esp\_now\_add\_peer(&peerInfo) != ESP\_OK) {

    Serial.println("Failed to add peer");

  } else {

    Serial.println("success add peer");

  }

  esp\_now\_register\_send\_cb(onDataSent);

  esp\_now\_register\_recv\_cb(esp\_now\_recv\_cb\_t(onDataReceive));

}

Untuk init esp-now, mengambil data mac address milik gateway yang tersimpan pada file system. Mac tersebut sebelumnya didapat pada proses pairing, untuk penjelasan pairing dapat dilihat pada document esp-now-gateway.

Kemudian melakukan init esp-now dan add peer milik gateway.