**گزارش فنی**

**جوی استیک بی سیم**

**گروه الکترونیک**

**شرکت فرند پیمایش (دانش بنیان)**

**شرکت فرند پیمایش (دانش بنیان)**

فهرست موضوعات

1. مقدمه 7

2. طرح مسأله و کاربرد مورد نظر 7

3. راه حل پیشنهادی 9

4. پیاده سازی الگوریتم 10

5. لیست توابع 14

6. روش استفاده از کتابخانه 16

7. پیوست 19

فهرست شکل‌ها

[شکل ‏1‑1 نام شکل. 9](#_Toc391910249)

فهرست جدول‌ها

[جدول 1پارامترهای مورد نظر در تولید Alarm 10](#_Toc131596355)

[جدول 2 متغیرهای بکار گرفته شده در فلوچارت Alarm 11](#_Toc131596356)

[جدول 3 توابع بکار گرفته شده در تولید Alarm 14](#_Toc131596357)

[جدول 4ساختارها و نمره گذاری‌ها 15](#_Toc131596358)

[جدول 5 ساختارها و نمره گذاری‌ها 15](#_Toc131596359)

قواعد نام گذاری:

متغیر ها و نام های توابع با استاندارد camelCase نام گذاری می شوند  
ثوابت ماکرو ها با استاندارد SCREAMING\_SNAKE\_CASE نام گذاری می شوند.

به دلیل اعمال تمایز استثنائاًTask ها به صورت camelCase\_task نام گذاری می شوند

# مقدمه

این گزارش برای معرفی نرم افزار و سخت افزار جوی استیک بلوتوثی پویشگر 3 تدوین شده است.

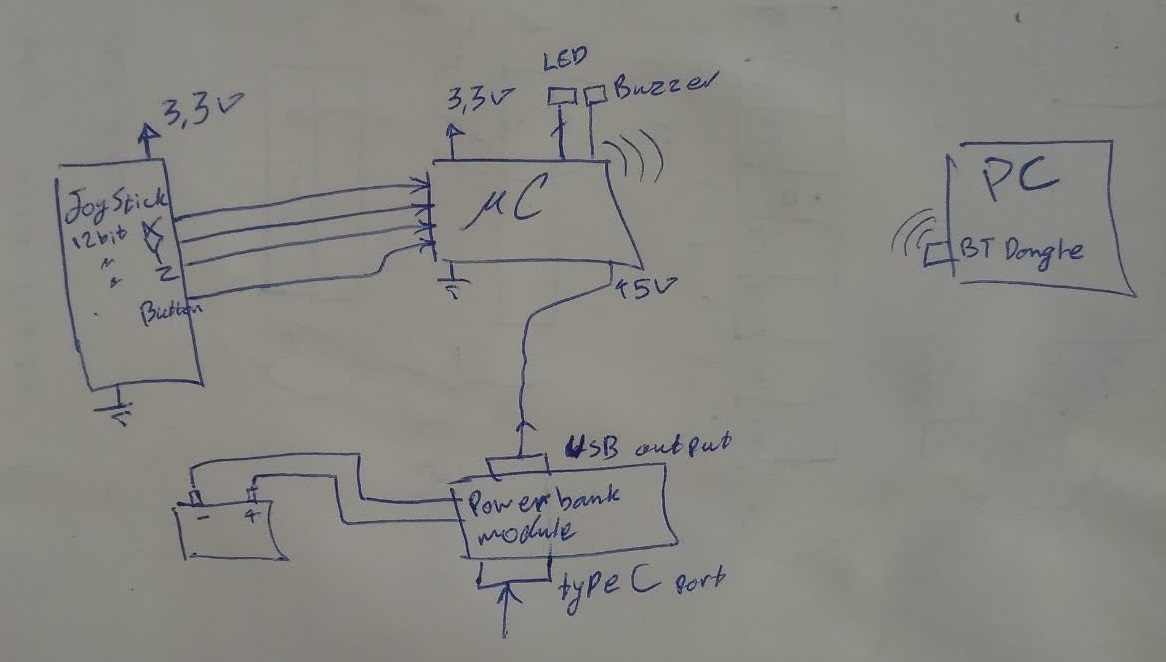
کارفرما برای هدایت راحت پویشگر، نیاز به کنترل کننده ای بی سیم دارد

# طرح مسأله و کاربرد مورد نظر

کنترل کننده باید دارای سه محور آنالوگ و یک یا دو دکمه باشد. میکروکنترلر باید بتواند این مقادیر را از جوی استیک خوانده و با ارتباط بلوتوثی آنها را به کامپیوتر مرکزی پویشگر ارسال کند. کنترل کننده دو چراغ LED برای نمایش وضعیت اتصال و وضعیت باتری خود دارد. همچنین، کیکروکنترلر باید بتواند دستوراتی مانند فعال سازی آلارم و روشن کردن LED و یا تنظیم پارامتر های خواندن اطلاعات مانند سرعت ارسال داده و... را از کامپیوتر دریافت کند.

# راه حل پیشنهادی

برای این دستگاه از میکروکنرلر ESP32، جوی استیک CHC-409B-M4 ، ماژول پاوربانک LX-LBU2C ، دو عدد باتری 18650 با ظرفیت 3200mAH ، یک دانگل بلوتوث Baseus BA04 استفاده می شود.



جدول 1: قطعات مورد استفاده

|  |  |
| --- | --- |
| Component | Description |
| Microcontroller | ESP32 Devkit Module |
| Analog Joystick | CHC-409B-M4 |
| Bluetooth Dongle | Baseus BA04 |
| Power Bank Module | LX-LBU2C |
| Battery | 2x 3200mAH 18560 |

جدول 1

# پیاده سازی سفت‌افزار

سفت افزار روی میکروکنترلر ESP32 اجرا شده که محیط برنامه نویسی آن پلتفرم Arduino است که روی FreeRTOS پیاده سازی شده.

از نرم افزار [Visual Studio Code](https://soft98.ir/software/programming/53-visual-studio-code-1.html) با افزونه PlatformIO برای توسعه سفت افزار استفاده شده است.

سفت افزار شامل چندین Task و Queue است که هر کدام باید به صورت Modular جدا از task های دیگر کار کنند.

### Task ها

FreeRTOS ، می تواند چندین Task با اولویت های متفاوت را همزمان اجرا کند.

### readJoystick\_task

این تسک، به صورت دوره ای مقادیر آنالوگ و کلید ها را خوانده و آنها به Queue تسک bluetoothManager\_task می فرستد..

### bluetoothManager\_task

این تسک، شامل شئ SerialBT از جنس BluetoothSerial است و مسئول ارسال و دریافت اطلاعات با بلوتوث است. ابتدا منتظر اتصال شده و سپس وارد حالت COMMAND\_MODE\_PACKET می شود و منتظر دریافت پارامتر های خواندن و ارسال اطلاعات می شود. پس از دریافت دستورات تنظیم اولیه، به صورت دوره ای اطلاعات آنالوگ و کلید ها را از Queue خود به نام qTransmitBT می خواند و آنها را با تابع constructByteArray تبدیل به یک آرایه از اعداد 8 بیتی کرده و آنها را به صورت سریالی از طریق بلوتوث ارسال می کند.

### alarm\_task

این تسک شامل کتابخانه Alarm\_FreeRTOS است و در کد اصلی برای استفاده از آلارم، تنها تابع addAlarmToQueue خوانده می شود.

بر خلاف سایر تسک های اصلی، این تسک در main.cpp شروع می شود.

### taskManager\_task

این تسک مسئول شروع تسک های دیگر با دستور serial و برای تست در مراحل اولیه نوشته شده است. برای کار خودکار این تسک مقادیر ADC\_TASK\_ENABLED و BT\_TASK\_ENABLED ست شده اند.

جدول 2

|  |  |  |
| --- | --- | --- |
| Task Names | Function | Description |
| readJoystick\_task | خواندن آنالوگ ها و کلید ها و ارسال به qTransmitBT |  |
| bluetoothManager\_task | خواندن آنالوگ ها و کلید ها از qTransmitBT و ارسال با بلوتوث |  |
| alarm\_task | اعمال آلارم ها |  |
| taskManager\_task | مدیریت تسک ها |  |

|  |  |
| --- | --- |
| Argument | Description |
| X Potentiometer  (Analog) |  |
| Y Potentiometer  (Analog) |  |
| Z Potentiometer  (Analog) |  |
| Key |  |
| LED |  |
| Buzzer |  |

# لیست توابع

جدول 3

|  |  |
| --- | --- |
| Description | Function |
| مقادیر دکمه ها را می خواند | readButtons |
| مقادیر آنالوگ را می خواند | readAxes |
| از یک MessageStruct، آرایه بایت را می سازد. می توان با تنظیم CODING\_0X1A، کد کردن 26 را فعال یا غیر فعال کرد. | constructByteArray |
| آرایه را کد می کند. | Code\_0x1A |

# ساختار­ها(Structures) و نمره­گذاری­های (Enumerations)

جدول 4

|  |  |  |
| --- | --- | --- |
| **Struct Name** | **Fileld** | **Description** |
| Joystick\_TypeDef | bool isConnected | وضعیت اتصال بلوتوث |
|  | uint8\_t mode | حالت عملیاتی را مشخص می کند |
|  | uint16\_t transmitRateMS | دوره تناوب خواندن و ارسال اطلاعات |
|  | float alpha | ضریب فیلتر پایین گذر |

|  |  |  |
| --- | --- | --- |
| **Struct Name** | **Fileld** | **Description** |
| MessageStruct | uint16\_t adc[] | مقادیر آنالوگ |
|  | bool button[] | مقادیر دکمه ها |

# روش ارتباط با کامپیوتر

7.1. تابع ساخت بسته اطلاعات:

void constructByteArray(MessageStruct \*message, byte \*arr)

{

int8 temp = 0;

for (int8 i = 0; i < BYTE\_ARRAY\_SIZE; i++)

arr[i] = 0;

for (int8 i = 0; i < ADC\_COUNT; i++)

{

arr[i \* 2 + 2] = message->adc[i] >> 8;

arr[i \* 2 + 3] = message->adc[i] & 0xFF;

}

arr[1] = message->button[0];

arr[COMMAND\_BYTE\_INDEX] = CMD\_START\_ACTION\_MODE;

if (CODING\_0X1A)

Code\_0x1A(arr, BYTE\_ARRAY\_SIZE);

{

که نتیجه آن به صورت زیر است:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| … | Command Data[0] | Command | ADC3L | ADC3H | ADC2L | ADC2H | ADC1L | ADC1H | Buttons | 0x1A? |
| … | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

## تنظیم پورت سریال در C#

## توابع دریافت داده از جویستیک

## توابع ارسال داده به جویستیک

بــرای بــه کــار بــردن ایــن کتـابـخـانـه در Main()، تــابـــع Configure\_Alarm() و در Timer\_Interrupt\_Handler() تابع UpdateAlarm() فراخوانی می­شود. لازم به ذکر است در صورتی که از نرم افزار MX Cube برای تنظیمات اولیه‌ی میکروکنترلر استفاده ‌شود، نیازی به فراخوانی تابع Configure\_Alarm() نمی‌باشد.

کد مثال زیر

Main()

{

…

Configure\_Alarm();

..

while(1)

{

}

}

Timer\_PeriodElapsed\_CallBack() // 32 Hz Interrupts

{

Update\_Alarm();

}

هر جا که ایجاد سوت مورد نیاز باشد، کافی است تابع Alarm به شکل زیر فراخوانی شود. مثلا:

…Alarm(SHORT\_BEEP\_X2, 3, 32, 1);…

در این تابع الگوی 2 سوت کوتاه با 3 تکرار هر کدام به طول 1 ثانیه فراخوانی شده است.

کد C ‏ ‏‏‏‏‏

# پیوست

## روش پروگرام کردن ESP32

### اتصال

### درایور USB to Serial

### ابزارهای نرم افزاری(Visual Studio Code Visual Studio C#)

### مراحل تغییر کد

* ادیت
* کامپایل
* آپلود ( فلش)

# عیب یابی

* (اگر LED اصلی چشمک نزند)
* (اگر LED شارژ به رنگ قرمز د بیاید)