F998 – Documentación Oficial v1.0

Esta documentación describe **completamente** el proyecto F998 en su versión 1.0:

- API del firmware Arduino
- Protocolo de comunicación serie
- Clase Python `F998`

El objetivo es disponer de una referencia **estable, clara y mantenible** para el desarrollo de aplicaciones.

---

1. API F998 (Firmware Arduino)

La API define las capacidades del hardware. Es independiente del protocolo serie y del software host.

1.1 Inicialización

```
void F998_setup();
```

Inicializa: - TM1628 - Matriz de LEDs - Teclado - Multiplexor de potenciómetros - Estado interno de LEDs y blink

Debe ser llamada desde `setup()`:

```
void setup() {
  F998_setup();
}
```

1.2 Entradas

Teclas

```
int tecla();
```

- Devuelve el código de la tecla pulsada (`9..39`)
- Devuelve `0` si no hay ninguna

Prioridad interna: 1. Teclado TM1628 2. Tecla directa (9) 3. Matriz externa

---

Potenciómetros

```
int pot(int a);
```

- `a = 1..9`
- Devuelve `0..100` (porcentaje)

```
int digPot(int a);
```

- `a = 1..9`
- Devuelve `0..6`
- El significado semántico se asigna en el host (por ejemplo `valor - 3`)

1.3 Matriz de LEDs / Vúmetros

```
void vumetro(int v, int porcentaje);
void vumetroR(int v, int porcentaje);
```

- `v = 0..3`
- Llenado horizontal izquierda→derecha o derecha→izquierda

```
void horizontalPos(int v, int c);
```

- Cursor horizontal
- `c = 0..8`

```
void verticalBar(int c, int porcentaje);
```

- Barra vertical
- Crecimiento desde abajo

```
void zoom(int c);
```

- Patrón triangular
- Columnas `0..c`
- Patrón `112233444`
- Crecimiento desde abajo
- Borra columnas superiores a `c`

```
void vumetroClear();
void vumetroFull();
```

1.4 Batería

```
void bateria(int porcentaje);
```

- Llenado de derecha a izquierda

```
void bateriaR(int porcentaje);
```

- Llenado de izquierda a derecha

```
void bateriaPos(int b, bool estado);
void bateriaBlink(int b, bool estado);
```

- `b = 1..4`
- Numeración humana e invertida

```
void bateriaClear();
```

1.5 LEDs de botones

```
void ledButton(int n, bool estado);
```

```
void ledInvert(int n);
```

- Invierte el **estado base** del LED

```
void ledBlink(int n, bool estado);
```

- Activa/desactiva blink sin modificar el estado base

1.6 Bajo nivel

```
void ledAt(int i, int j, bool estado);
```

- `i = grid (0..6)`
- `j = segmento (0..9)`

```
void ledAtBlink(int i, int j, bool estado);
```

1.7 Animaciones

```
void animModo(uint8_t mask);
```

- `mask` usa bits 0..3 para seleccionar vúmetros
- Animación bloqueante

1.8 Blink

```
void updateBlink();
```

Debe ejecutarse periódicamente desde `loop()`.

---

2. Protocolo de Comunicación Serie

2.1 Capa física

- USB Serial
- 115200 baudios
- ASCII
- `\n` como terminador

Formato general:

```
COMANDO arg1 arg2 ...
```

2.2 Comandos de lectura

Identificación

```
ID?
```

Respuesta:

```
F998 v1.0
```

Estado completo

```
GET
```

Respuesta:

```
S P=34,78,12,90,0,0,0,0,0 D=3,4,2,3,3,3,3,3,3
```

Polling rápido de tecla

```
IN
```

Respuesta:

```
K <n>
```

- `<n> = 0` → ninguna tecla

2.3 LEDs de botones

```
LB  n 0|1
LBB n 0|1
LBI n
```

2.4 Matriz / vúmetros

```
VU  v p
VUR v p
HP  v c
VB  c p
ZM  c
VC
VF
```

2.5 Batería

```
BAT  p
BATR p
```

```
BATP b 0|1
BATB b 0|1
BATC
```

2.6 Bajo nivel

```
LAT i j 0|1
LAB i j 0|1
```

2.7 Animación

```
AN mask
```

2.8 Errores

```
ERR 1
```

---

3. Clase Python `F998`

3.1 Dependencias

```
pip install pyserial
```

3.2 Filosofía

- Python controla la lógica
- Arduino ejecuta
- Sin estados ocultos
- Comunicación sincrónica

3.3 Constructor

```
F998(port, baud=115200, timeout=0.05)
```

3.4 Lectura

```
f.id()
f.in_key()
f.get()
```

3.5 LEDs de botones

```
f.led(n, True)
f.led_blink(n, True)
f.led_invert(n)
```

3.6 Matriz / vúmetros

```
f.vumetro(v, p)
f.vumetro_r(v, p)
f.vertical_bar(c, p)
f.zoom(c)
f.clear_vu()
f.full_vu()
```

3.7 Batería

```
f.battery(p)
f.battery_r(p)
f.battery_pos(b, True)
f.battery_blink(b, True)
f.battery_clear()
```

3.8 Bajo nivel

```
f.led_at(i, j, True)
f.led_at_blink(i, j, True)
```

3.9 Animación

```
f.anim(mask)
```

3.10 Cierre

```
f.close()
```

---

4. Estado del proyecto

- API firmware estable
- Protocolo cerrado
- Cliente Python funcional
- Base sólida para desarrollo de aplicaciones

Fin del documento – F998 v1.0