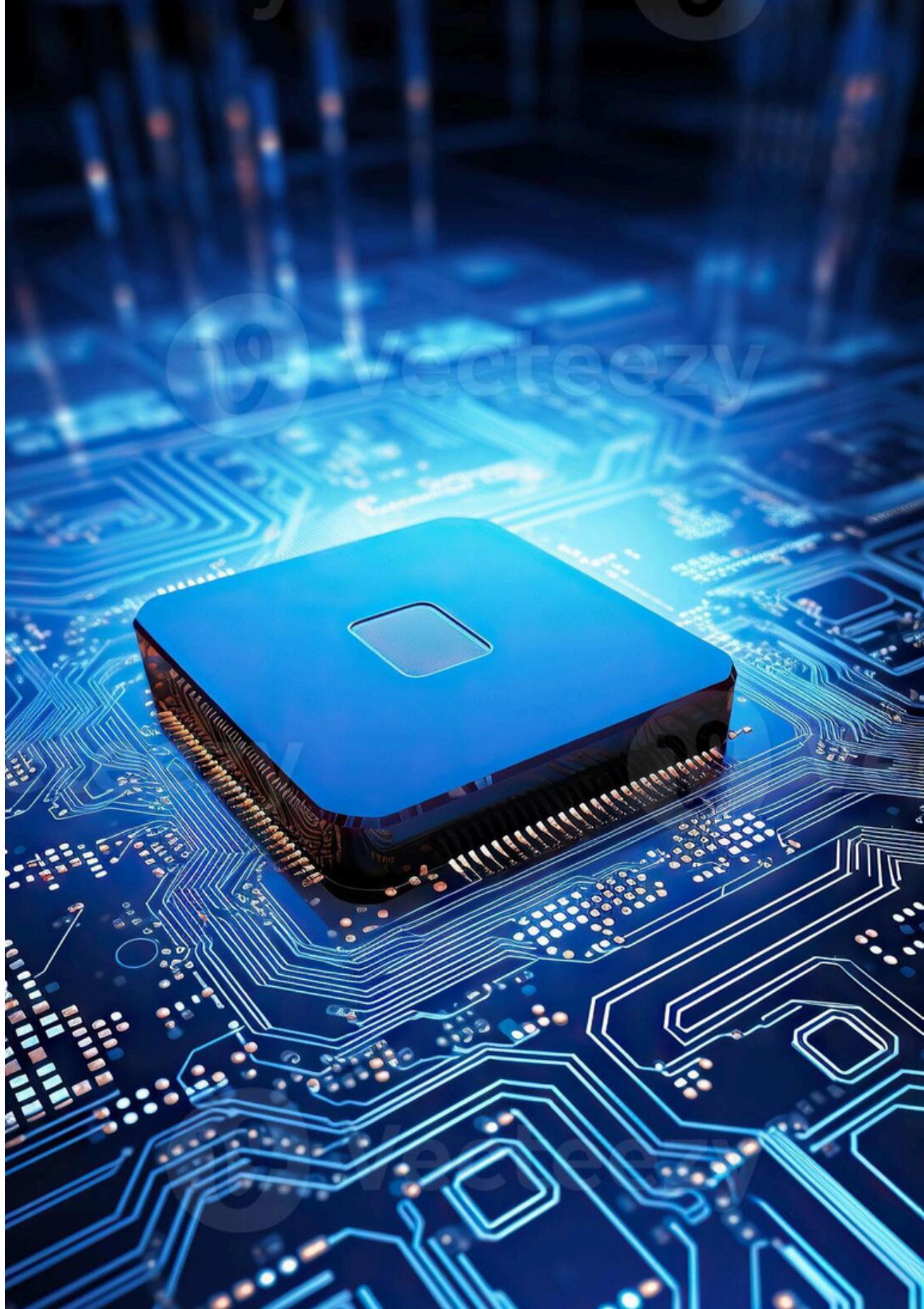


# **Elaborato di Architettura e Progetto dei Calcolatori**

Corso del Professor Mario Barbareschi - Anno 2023/24

**ALESSIO COCCA M63001692**  
**SIMONE DI FRAIA M63001678**





# Cassaforte di Sicurezza

## Funzionalità 1

Apertura e chiusura automatica  
tramite autenticazione RFID

## Funzionalità 2

Antifurto legato alla prolungata  
vicinanza alla cassaforte

# Dispositivi Utilizzati

1

**Lettore RFID RC522**

2

**Servomotore SG90**

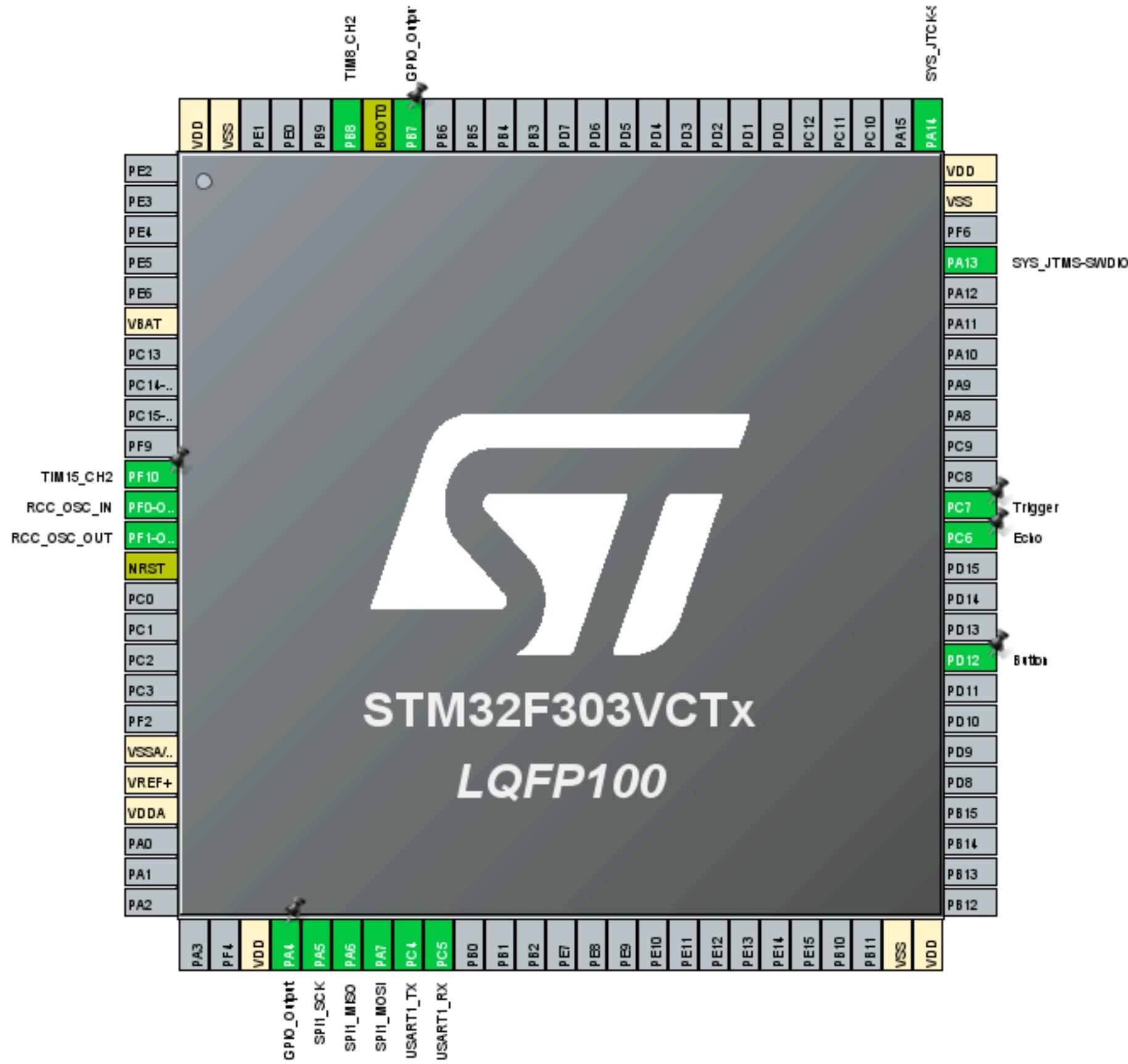
3

**Sensore di distanza a ultrasuoni HC-SR04**

4

**Cicalino Piezoelettrico KY-006**





RFID

- PB7: Reset
  - PA4: Chip Select
  - PA5: Serial Clock
  - PA6: MISO
  - PA7: MOSI

HC-SR04

- PC7: Trigger
  - PC6: Echo

SG90

- PB8: Timer 8

KY-006

- PF10: Timer 15

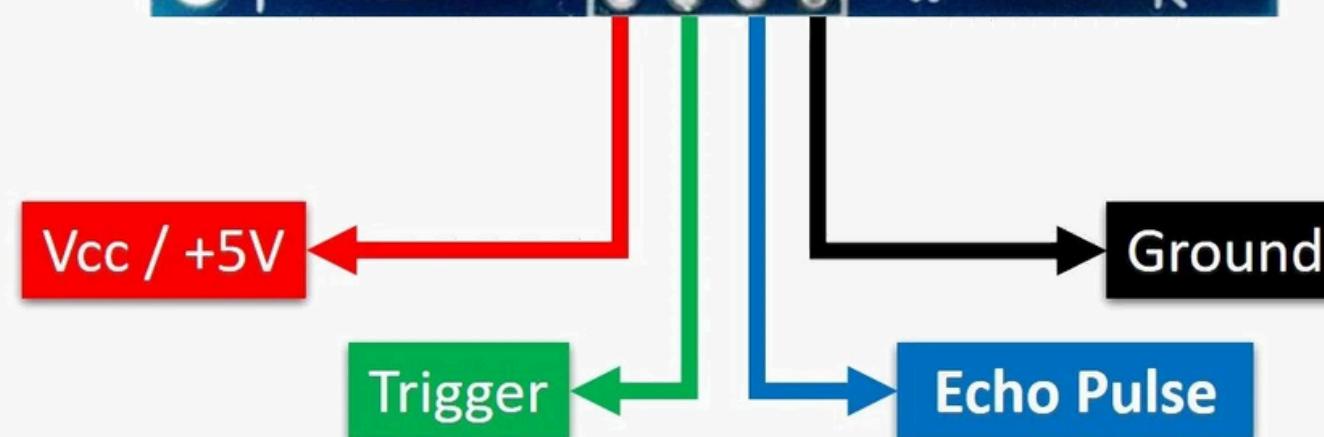
# Button

# HC-SR04

Sensore a ultrasuoni utilizzato per misurare la distanza tra un oggetto e il sensore stesso



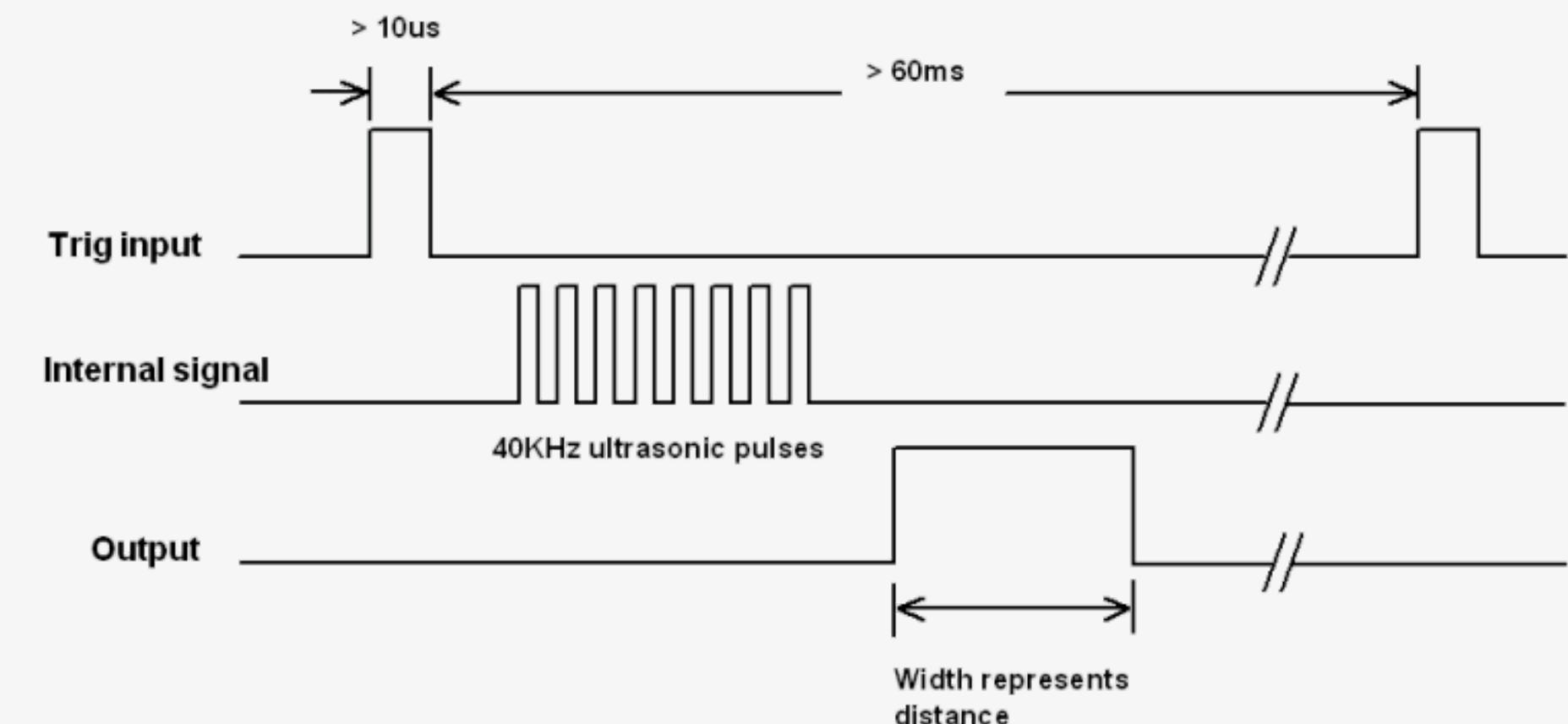
- Gamma di Rilevamento: 2cm a 400cm
- Frequenza di lavoro: 40 kHz
- Alimentazione: 5V DC
- Interfaccia: 4 pin (VCC, GND, TRIG, ECHO)



# HC-SR04



1. Il pin TRIG invia un impulso di 10 microsecondi.
2. Il sensore trasmette un segnale ultrasonico.
3. Quando l'onda si riflette su un ostacolo, viene ricevuta dal sensore e il pin ECHO restituisce un segnale proporzionale al tempo impiegato dall'onda per tornare.
4. La distanza in centimetri viene calcolata come il tempo (in microsecondi) per cui ECHO resta alto diviso 58.



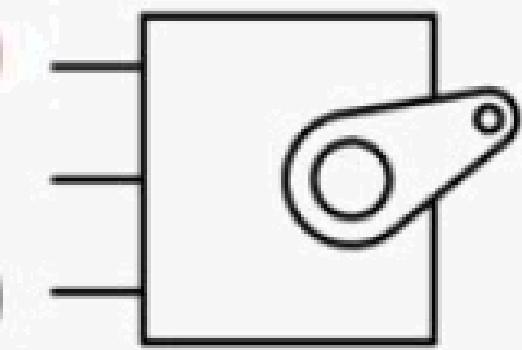


## SG90

Servomotore a rotazione limitata, compatto e leggero

- Angolo di rotazione: 0° - 180°.
- Coppia: 2.5 kg/cm.
- Alimentazione: 4.8V - 6V DC.
- Dimensioni: 23 x 12 x 32 mm.
- Peso: 14.7 g
- Interfaccia: 3 pin (VCC, GND, Segnale).

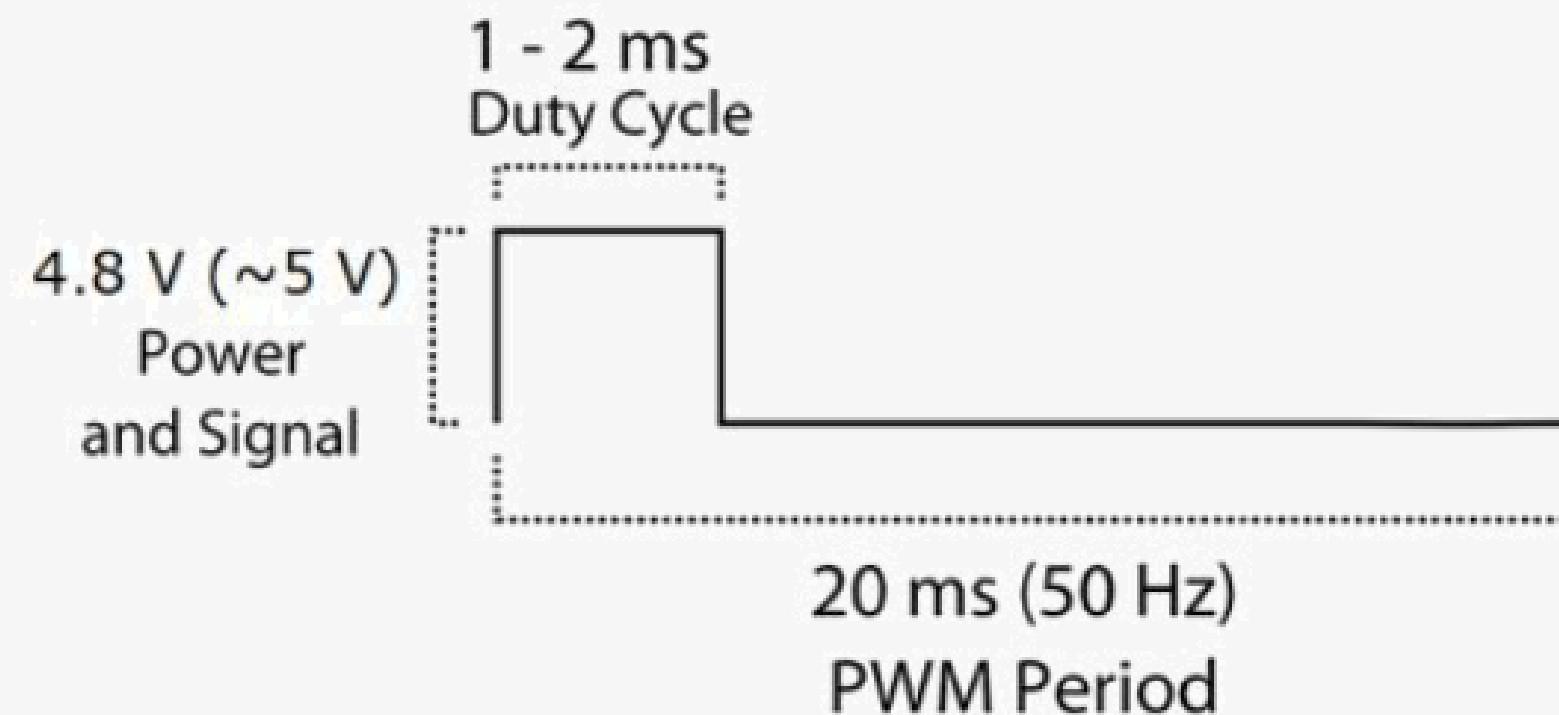
PWM=Orange (⊜⊜)  
Vcc= Red (+)  
Ground=Brown (-)





## SG90

1. Controllo tramite segnale PWM (Pulse Width Modulation).
2. Il segnale di controllo determina la posizione dell'albero del servo:
  - Impulsi con durata 1.5ms (posizione 0°)
  - Impulsi con durata di circa 2ms (posizione 90°)
  - Impulsi con durata di circa 1 ms (posizione -90°)

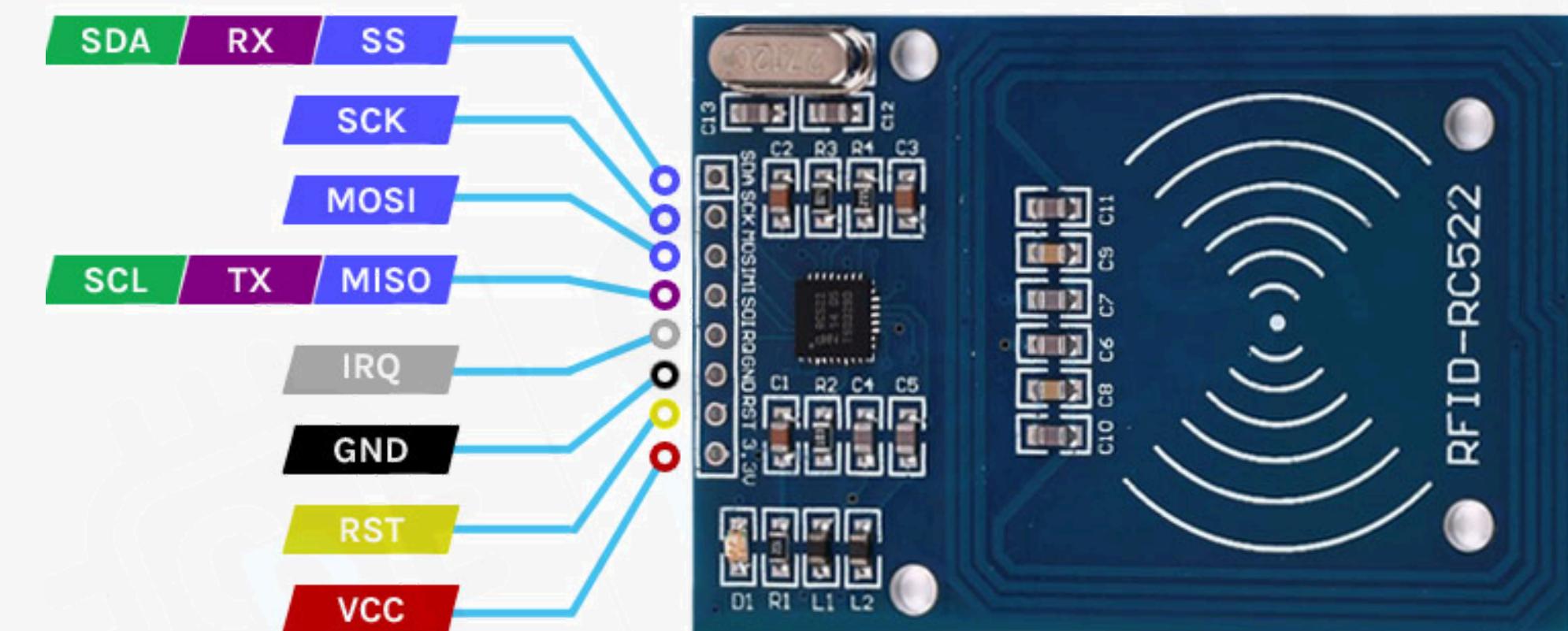


# RFID RC522

Modulo a radiofrequenza (RFID) utilizzato per leggere e scrivere dati su tag RFID a 13.56 MHz

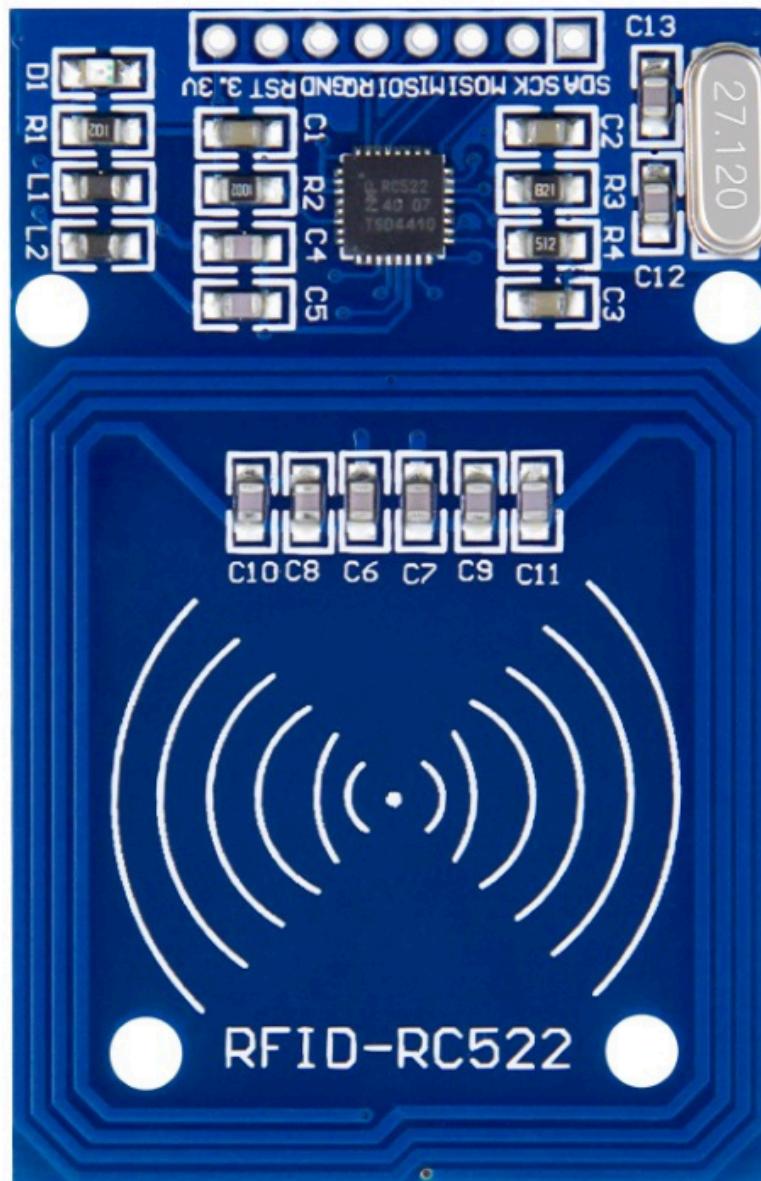


- Basato sul chip MFRC522, compatibile con schede e tag RFID della serie ISO/IEC 14443
- Interfaccia: SPI (Serial Peripheral Interface), con supporto anche per I<sup>2</sup>C e UART
- Distanza di lettura: Fino a 10 cm
- Alimentazione: 3.3V DC
- Dimensioni: 40mm x 60mm
- Consumo energetico: Basso consumo in modalità stand-by (meno di 80 µA)

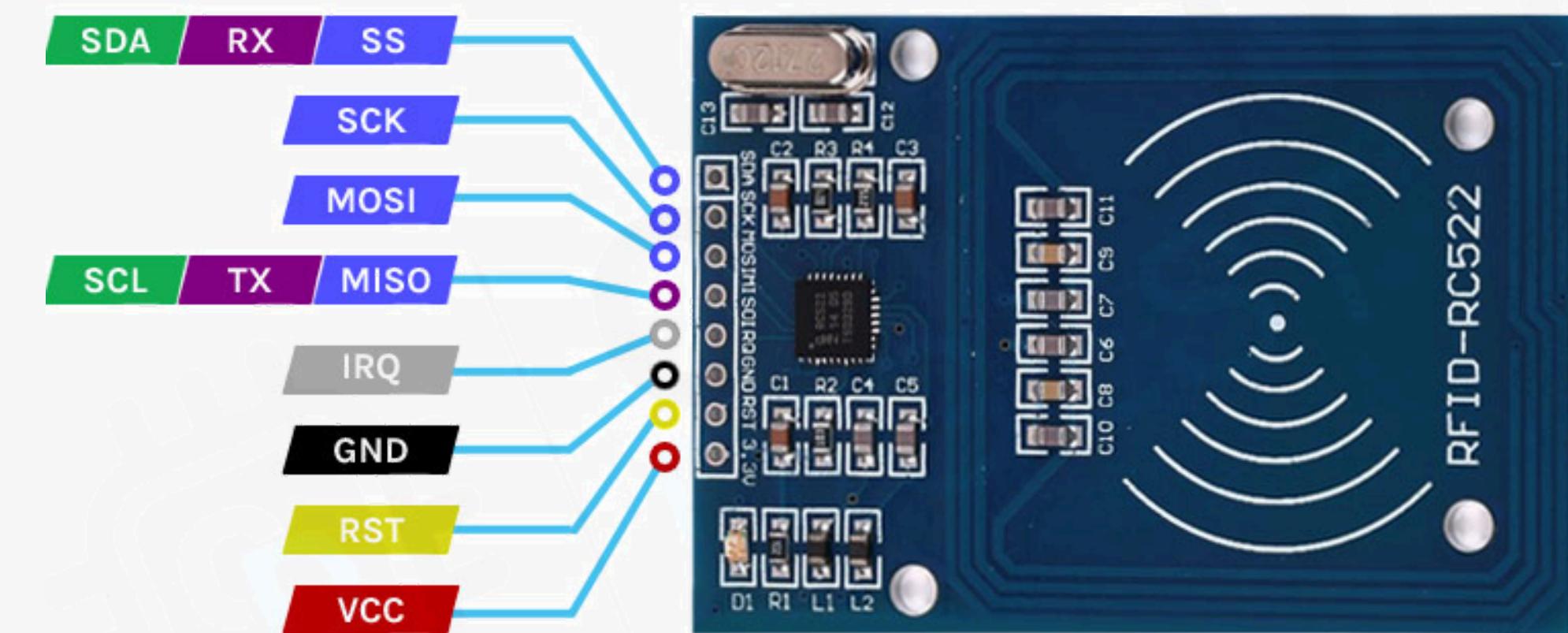


# RFID RC522

Modulo a radiofrequenza (RFID) utilizzato per leggere e scrivere dati su tag RFID a 13.56 MHz



1. Il modulo comunica con il microcontrollore tramite l'interfaccia SPI.
2. Utilizza la tecnologia Near Field Communication (NFC) per scambiare dati tra il lettore e i tag.
3. I tag RFID sono passivi, attivati dalla potenza elettromagnetica emessa dal modulo.
4. Le operazioni di lettura e scrittura vengono gestite tramite appositi comandi inviati dal microcontrollore

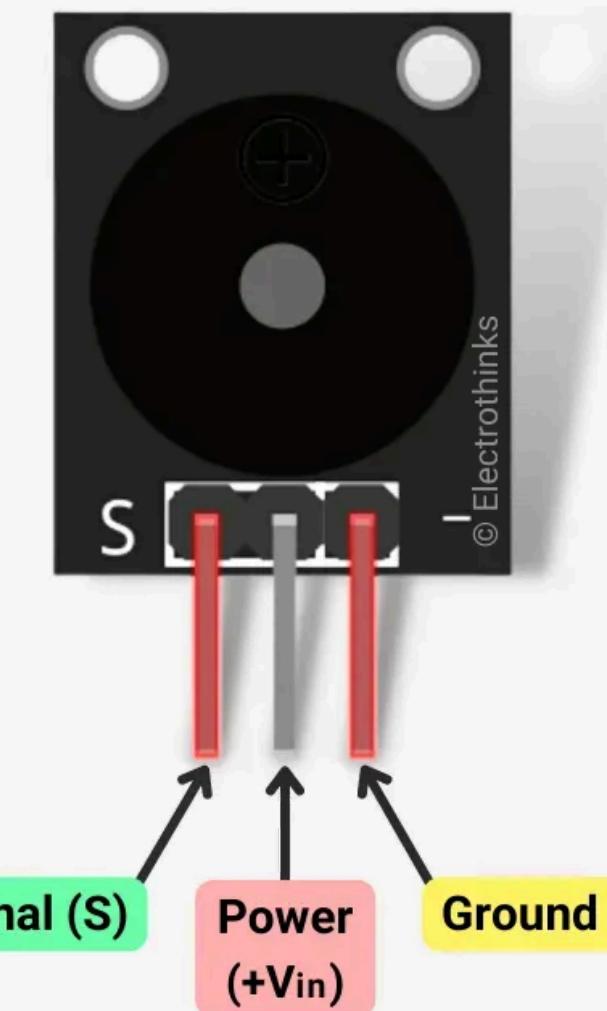


# KY-006

Modulo piezoelettrico passivo utilizzato per generare segnali acustici tramite un segnale elettrico.



- Tensione operativa: 1.5V - 15V DC
- Frequenza sonora: 1.5 a 2.5 kHz dipendente da un segnale di controllo
- Interfaccia: 3 pin (VCC, GND, Segnale)
- Dimensioni: 18.5mm x 15mm



# KY-006

Modulo piezoelettrico passivo utilizzato per generare segnali acustici tramite un segnale elettrico.



- Il cicalino passivo non genera suono autonomamente, ma richiede un segnale di controllo in ingresso che generiamo grazie al microcontrollore.
- Il suono prodotto dipende dalla frequenza del segnale PWM applicato al pin di controllo.
- Più alto è il valore del PWM, più alta sarà la frequenza del suono emesso

**Funzionalità 1**

# **Apertura e chiusura automatica tramite autenticazione RFID**



## Funzionalità 1

Apertura e chiusura automatica  
tramite autenticazione RFID

# Passi da eseguire

1

Attiviamo il lettore RFID per  
renderlo in attesa di comunicare  
con un Tag RFID

2

Non appena viene avvicinato  
un Tag RFID, leggiamo  
l'identificativo corrispondente

3

Se l'identificativo coincide con  
quello indicato, viene attivato il  
servomotore che procede ad  
aprire/chiudere la cassaforte

## Abilitazione del modulo RC522 e lettura del Tag RFID

1

In primo luogo, andiamo a inizializzare il modulo RC522. Dopo aver resettato il modulo, aver settato i timer interni, configurato la modalità di trasmissione e i registri per il CRC, l'inizializzazione prevede di abilitare l'antenna per poter comunicare con i Tag.

```
void MFRC522_Init(void)
{
    HAL_GPIO_WritePin(MFRC522_CS_PORT, MFRC522_CS_PIN, GPIO_PIN_SET);
    HAL_GPIO_WritePin(MFRC522_RST_PORT, MFRC522_RST_PIN, GPIO_PIN_SET);
    MFRC522_Reset();

    //Timer: TPrescaler*TreloadVal/6.78MHz = 24ms
    Write_MFRC522(TModeReg, 0x8D);           //Tauto=1; f(Timer) = 6.78MHz/TPreScaler
    Write_MFRC522(TPrescalerReg, 0x3E);      //TModeReg[3..0] + TPrescalerReg
    Write_MFRC522(TReloadRegL, 30);
    Write_MFRC522(TReloadRegH, 0);

    Write_MFRC522(TxAutoReg, 0x40);          // force 100% ASK modulation
    Write_MFRC522(ModeReg, 0x3D);           // CRC Initial value 0x6363

    AntennaOn();
}
```

Nota: le funzionalità riportate sono state fornite dal produttore del dispositivo in librerie importabili in codice. Nel codice è stata quindi importata la libreria "rc522.h", specifica per l'uso del dispositivo con le schede stm32.

2

## Abilitazione del modulo RC522 e lettura del Tag RFID

In un ciclo continuo, richiamiamo le funzioni MFRC522\_Request e MFRC522\_Anticoll: la prima consente di inviare un comando al modulo per cercare Tag RFID nelle vicinanze e se esso è stato rilevato correttamente, il tipo di tag viene memorizzato nell'array str, la seconda invece viene impiegata per implementare un algoritmo di anticollisione per gestire la situazione in cui più tag RFID sono presenti nel raggio del lettore. Se la procedura, in questione, va a buon fine il numero seriale del Tag viene riportato nell'array str.

```
uchar MFRC522_Request(uchar reqMode, uchar *TagType)
{
    uchar status;
    uint backBits;           // The received data bits

    Write_MFRC522(BitFramingReg, 0x07);      //TxLastBists = BitFramingReg[2..0]

    TagType[0] = reqMode;
    status = MFRC522_ToCard(PCD_TRANSCEIVE, TagType, 1, TagType, &backBits);

    if ((status != MI_OK) || (backBits != 0x10))
    {
        status = MI_ERR;
    }

    return status;
}

uchar MFRC522_Anticoll(uchar *serNum)
{
    uchar status;
    uchar i;
    uchar serNumCheck=0;
    uint unLen;

    Write_MFRC522(BitFramingReg, 0x00);      //TxLastBists = BitFramingReg[2..0]

    serNum[0] = PICC_ANTICOLL;
    serNum[1] = 0x20;
    status = MFRC522_ToCard(PCD_TRANSCEIVE, serNum, 2, serNum, &unLen);

    if (status == MI_OK)
    {
        //Check card serial number
        for (i=0; i<4; i++)
        {
            serNumCheck ^= serNum[i];
        }
        if (serNumCheck != serNum[i])
        {
            status = MI_ERR;
        }
    }

    return status;
}
```

Nota: le funzionalità riportate sono state fornite dal produttore del dispositivo in librerie importabili in codice. Nel codice è stata quindi importata la libreria "rc522.h", specifica per l'uso del dispositivo con le schede stm32.

## Attivazione del meccanismo di Apertura/Chiusura tramite servomotore

Se il numero seriale del Tag corrisponde a quello del Tag posseduto dal proprietario della cassaforte, allora si mette in moto il meccanismo di apertura e chiusura della cassaforte.

Si sfrutta un flag per poter distinguere tra i due diversi stati della cassaforte (aperta, chiusa).

Come il cicalino, anche il servomotore necessita di un segnale di controllo PWM generato dal microcontrollore, per poter gestire le diverse posizioni assunibili.

A seconda dello stato della cassaforte, andiamo a controllare il duty cycle in modo tale da consentire al servomotore di poter girare di 180 gradi verso sinistra e verso destra per controllare l'apertura e chiusura di essa.

Nota: le funzionalità riportate sono state fornite dal produttore del dispositivo in librerie importabili in codice. Nel codice è stata quindi importata la libreria "rc522.h", specifica per l'uso del dispositivo con le schede stm32.

```
while (1)
{
    if(!MFRC522_Request(PICC_REQIDL, str)){
        if(!MFRC522_Anticoll(str)){
            printf("UID CARD: %u %u %u %u %u \r\n", str[0], str[1], str[2], str[3], str[4]);
            if(str[0] == 163 & str[1] == 49 & str[2] == 146 && str[3] == 13){
                if(opened == 0){
                    opened = 1;
                    HAL_TIM_SET_COMPARE(&htim8, TIM_CHANNEL_2, Right);
                    HAL_Delay(20);
                }
                else{
                    opened = 0;
                    HAL_TIM_SET_COMPARE(&htim8, TIM_CHANNEL_2, Left);
                    HAL_Delay(20);
                }
            }
        }
    }
}
```

**Funzionalità 2**

# **Antifurto legato alla prolungata vicinanza alla cassaforte**



## Funzionalità 2

Antifurto legato alla prolungata vicinanza alla cassaforte

# Passi da eseguire

1

Misurazione della distanza degli oggetti di fronte al sensore

2

Avvio antifurto se oggetti sono distanti meno di 40cm per più di 10 secondi

3

Eventuale disattivazione dell'antifurto tramite pressione di un pulsante

## Misurazione della distanza tramite sensore HC-SR04

1

Sfruttiamo il Timer 4 per poter alzare il segnale di Trigger per 10 microsecondi: alziamo il livello logico della linea di Trigger e dopo il periodo trascorso viene generata un'interruzione per abbassare il livello logico del pin.

```
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_SET);  
HAL_TIM_SetCounter(&htim4, 0);  
HAL_TIM_Base_Start_IT(&htim4);
```

```
@void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)  
{  
    if(htim == &htim4){  
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_RESET);  
        HAL_TIM_Base_Stop_IT(&htim4);  
    }  
}
```

## Misurazione della distanza tramite sensore HC-SR04

2

Utilizziamo nuovamente il meccanismo delle interruzioni per misurare la durata in cui il segnale Echo resta alto, dato con cui poi possiamo andare a misurare la distanza e verificare se essa sia inferiore o uguale ai 40 cm.

Generiamo, dunque, un'interruzione sia sul fronte di salita del segnale Echo che ci consente di avviare il conteggio del Timer 3, sia sul fronte di discesa per ottenere il valore del conteggio finale.

Dividendo il valore di conteggio ottenuto per 58 otteniamo la distanza in centimetri. Se essa è inferiore o uguale a 40 cm, incrementiamo una variabile contatore.

```
if(GPIO_Pin == Echo_Pin){  
    if(HAL_GPIO_ReadPin(Echo_GPIO_Port, Echo_Pin) == GPIO_PIN_SET){  
        HAL_TIM_Base_Start(&htim3);  
    }else{  
        HAL_TIM_Base_Stop(&htim3);  
        count = __HAL_TIM_GET_COUNTER(&htim3);  
        distanza = count/58;  
        __HAL_TIM_SET_COUNTER(&htim3, 0);  
  
        if(distanza <= 40){  
            count_wait++;  
        }  
        else{  
            count_wait = 0;  
        }  
  
        printf("Echo Basso. Tick: %u. Distanza: %u. Count: %u \r\n ", currentMs, distanza, count_wait);  
    }  
}
```

## Avvio Antifurto

Quando il contatore, legato alla distanza, raggiunge il valore 10, viene richiamata la procedura di avvio dell'antifurto.

La procedura di avvio fa uso del Timer 15. Il cicalino necessita di un segnale di controllo esterno PWM. Il Timer 15 viene opportunamente settato in modalità PWM generator facendo sì che generi un segnale con frequenza 2kHz (tipica degli allarmi).

Nella procedura di avvio antifurto, andiamo a regolare il duty cycle del segnale in questione, in modo tale che il cicalino emetta un suono intermittente.

```
if (count_wait == 10) {
    avvia_antifurto();
}

void avvia_antifurto() {

    while (button_flag == 0) {
        HAL_TIM_SET_COMPARE(&htim15, TIM_CHANNEL_2, 500);
        HAL_Delay(100);

        HAL_TIM_SET_COMPARE(&htim15, TIM_CHANNEL_2, 0);
        HAL_Delay(100);
    }
}
```

## Disattivazione Antifurto tramite pressione del Pulsante

Esiste la possibilità di disattivare l'antifurto, sia quando esso è in azione (per rinuovere il rumore fastidioso) sia quando l'utente legittimo decide di fare un utilizzo prolungato della cassaforte.

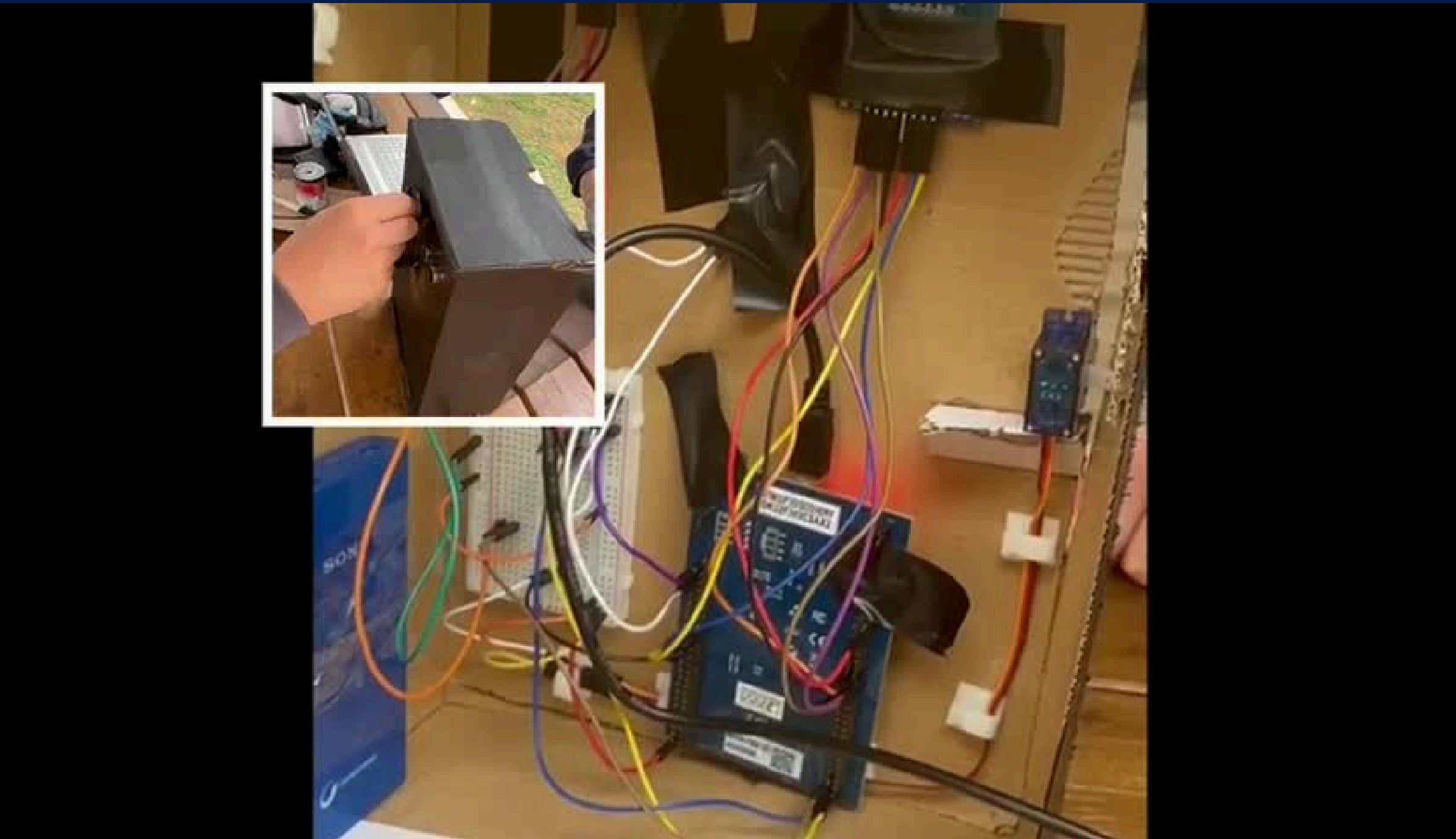
Il pulsante è della tipologia Switch Button, quindi nel momento in cui si effettua pressione di esso, la linea logica collegata a uno dei suoi terminali si abbassa in quanto lo switch fa sì che il pin sia collegato a massa. Sfruttiamo infatti la resistenza interna di Pull-Up, abilitando la relativa modalità delle GPIO, che è tale per cui quando il pulsante non è premuto, il pin sia connesso al livello logico alto. Il pin potrebbe altrimenti restare in uno stato indeterminato (fluttuante) e non assicurare una lettura stabile.

Si sfrutta un flag: esso è pari a 1 quando l'antifurto è disattivato, pari a 0 nel caso opposto. Quando premo il pulsante passo da uno stato all'altro.

Si rende necessario il conteggio dei millisecondi di distanza tra due interruzioni legate al pulsante per ragioni legati al Debouncing del pulsante.

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){  
    currentMs = HAL_GetTick();  
  
    if(GPIO_Pin == Button_Pin && (currentMs - previousMs) > 300){  
  
        if(button_flag == 0){  
            printf("Ho disattivato l'antifurto: %u \r\n", currentMs - previousMs);  
            button_flag = 1;  
        }  
        else{  
            printf("Ho riattivato l'antifurto %u \r\n", currentMs - previousMs);  
            button_flag = 0;  
            count_wait = 0;  
        }  
  
        previousMs = currentMs;  
    }  
}
```

## Video 1: Apertura e chiusura automatica tramite autenticazione RFID



## **Video 2: Antifurto legato alla prolungata vicinanza alla cassaforte**



Grazie per l'attenzione!