Night Surveillance System: termocamera di sorveglianza notturna basata su STM32F407.

Progettazione di Sistemi Operativi

Università degli Studi di Milano.

July 12, 2022

Descrizione

Il dispositivo si propone come prototipo di camera di sorveglianza notturna con le seguenti modalità di funzionamento:

1. Manuale: consente acquisizione e streaming immagine termica, acquisizione e playback audio ambientale in stereo e orientamento manuale della camera tramite joystick.
2. Follow: il dispositivo reagisce a stimoli sonori sopra soglia cercando di localizzare una sorgente sonora nello spazio circostante.

Componenti:

1. Camera termica: AMG8833
2. Microfoni a elettrete
3. Motore stepper unipolare 28BYJ-48
4. Joystick analogico
5. Bottoni di controllo
6. STM32F407-discovery1

Elementi d'interesse

L'obiettivo del progetto è quello di realizzare un sistema di sorveglianza che monitori diversi tipi di dato ambientali e che possa reagire attivamente a cambi di circostanze nel perimetro da monitorare.
Elementi d'interesse a livello di progettazione software:

1. Gestione simultanea di streaming real-time multipli e controllo attuatori.
2. Interazione a basso livello con le componenti hardware e sviluppo di driver dedicati.
3. Scrittura di ISR, utilizzo DMA e periferiche senza mediazione di un RTOS.

Camera termica

Griglia di sensori IR (Grid-EYE). Panasonic AMG8833 specifiche:

1. 64 px
2. 10 fps o 1 fps
3. I2C slave interface fast mode dual address (7 bit)
4. INT pin
5. Interrupt absolute/hysteresis mode

Per leggere i dati dalla camera e cambiare i suoi parametri operativi è sufficiente leggere/scrivere i registri interni secondo le modalità indicate sul datasheet.

TITLE	SPECIFICATIONS FOR Infrared Array Sensor	PAGE 14/26
NAME	Infrared Array Sensor "Grid-EYE"	AMG88**

(11) Temperature Register

Register for reading only to indicate temperature data per 1 pixel.
Temperature Data of each pixel is 12 bit data and 2 byte data.
1 LSB has 12 bit resolution (11 bit + sign) which is equivalent to 0.25℃
and it is indicated as two's complement form.
Main temperature data are shown below.

address	register	R/W	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0x80	T01L	R	T7	T6	T5	T4	T3	T2	T1	T0	0x00
0x81	T01H	R	-	-	-	-	+/-	T10	T9	T8	0x00

temperature	Binary number	HEX number
+125℃	0001_1111_0100	0x1F4
+25℃	0000_0110_0100	0x064
+0.25℃	0000_0000_0001	0x001
0℃	0000_0000_0000	0x000
-0.25℃	1111_1111_1111	0xFFF
-25℃	1111_1001_1100	0xF9C
-55℃	1111_0010_0100	0xF24

Figure 1: Specifiche registri di temperatura datasheet Grid-EYE AMG8833

Microfoni a elettrete

Microfoni a condensatore basati su membrana di elettrete con carica elettrostatica permanente (no polarizzazione esterna). Alimentazione necessaria per transistor FET di adattamento in impedenza.
Specifiche:

1. Range di risposta in frequenza: 20-20kHz.
2. Ampiezza picco-picco 200mV ($\pm 100mV$) in saturazione.

Preamplificazione: condizionamento necessario per portare il segnale nel range ADC [0V-3V]. Rail-to-rail opamp (TLE2022) in configurazione invertente con partitore su terminale positivo.

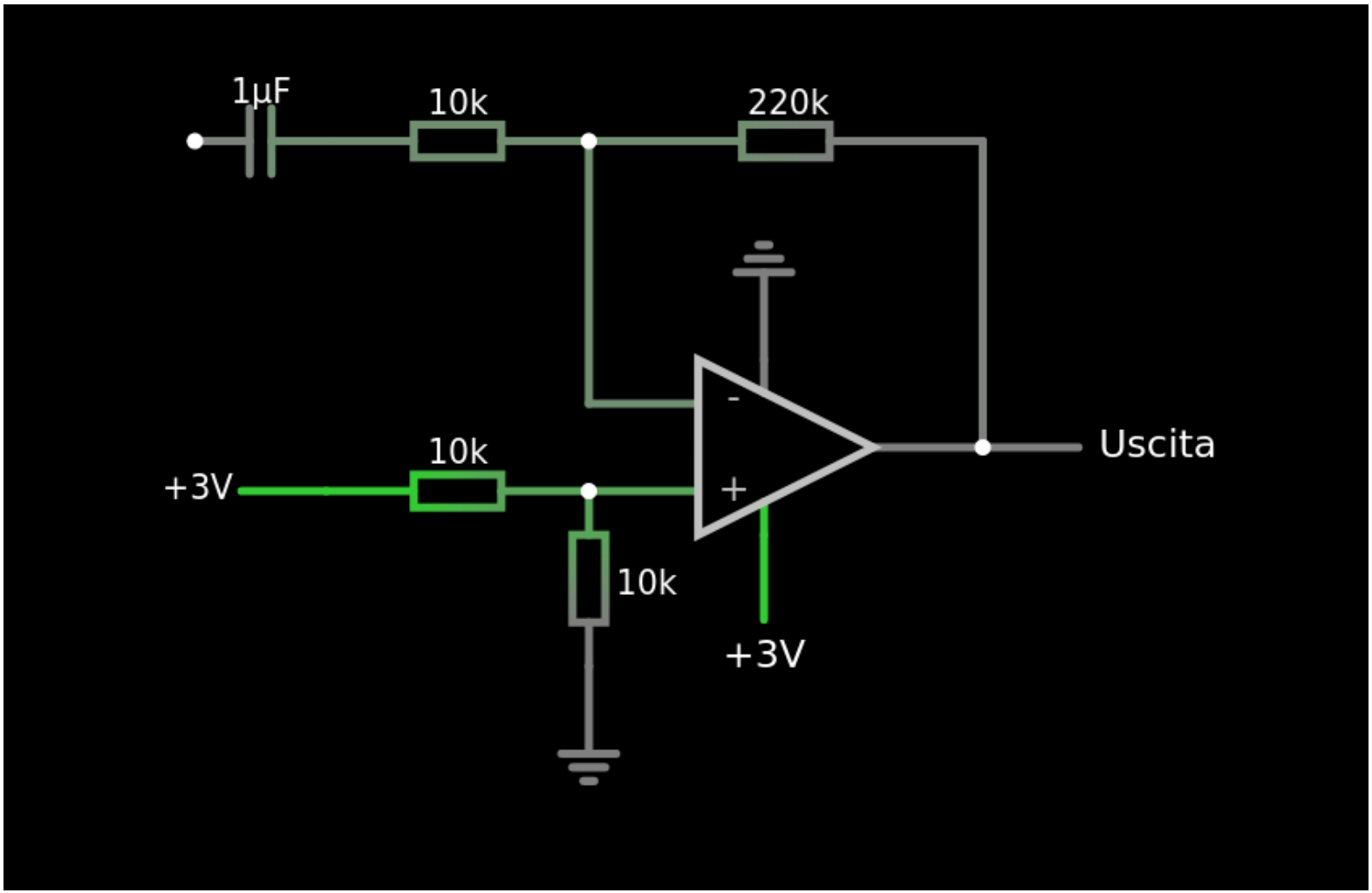


Figure 2: Schema circuitale preamplificatore microfonico

Motore stepper e joystick

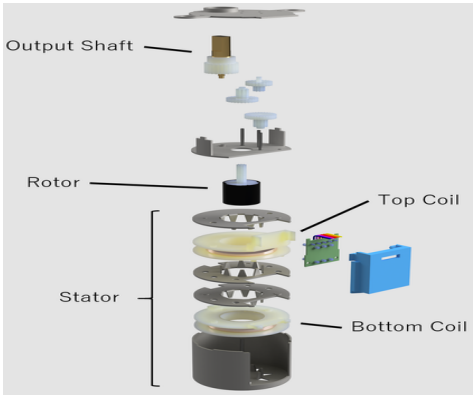


Figure 3: Vista esplosa 24BYJ-48

Specifiche 24BYJ-48:

1. Risoluzione rotore $= 11.25°$.
2. Gear ratio $= 1/63.68$.
3. Step per rivoluzione $= 2038$.
4. Chip driver: ULN2003.

Joystick == potenziometro.

I2C

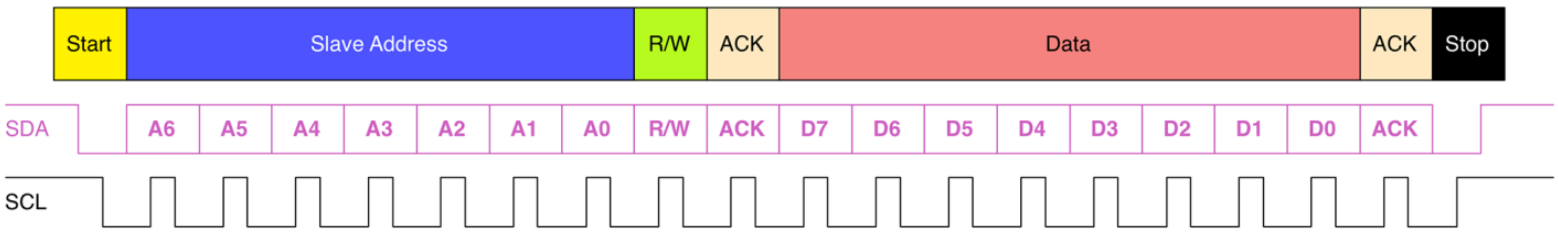


Figure 4: Protocollo di comunicazione seriale I2C

Master inizia le transazioni di lettura/scrittura tramite START condition (HIGH to LOW). Specifica indirizzo dello slave in address frame. Il primo data frame specifica l'indirizzo del registro da cui leggere/in cui scrivere e gli altri sono di dati. Lettura/scrittura con ACK fino a STOP condition (da LOW ad HIGH).
Sensore AMG8833 configurato per rispondere a indirizzo a 7 bit 0x68. PB6 e PB7 con resistenze di pull-up esterne per line SCL e SDA a 400kbps.

AMG8833 Driver

```
1  void amg8833Init(AMG8833 *inst,
       I2C_HandleTypeDef *hi2c,uint8_t
       ad_sel){
2    inst->adri2c= ad_sel ? (
       AMG8833_I2C_BASE_ADR + 1 ) << 1 :
       AMG8833_I2C_BASE_ADR << 1;
3    inst->hi2c=hi2c;
4  }
```

```
1  /*
2   * Read temperature matrix in DMA mode
3   */
4  HAL_StatusTypeDef amg8833ReadDMA(AMG8833 *
       inst,uint8_t *data, uint8_t max_retry
       ){
5
6    return I2C_ReadDMA( inst,AMG8833_T01L,
       data,AMG8833_DS,max_retry );
7  }
8
```

```
1  /*
2   * Utils to read/write on I2C line with
       automatic retry in case of failure
3   */
4  HAL_StatusTypeDef I2C_Read(AMG8833 *inst,
       uint8_t reg,uint8_t *data,uint16_t
       dim,uint8_t max_retry){
5
6    uint8_t try=0;
7    HAL_StatusTypeDef status;
8
9    do{
10     try++;
11     status=HAL_I2C_Mem_Read_DMA( inst
       ->hi2c,(uint16_t)inst->adri2c,reg,1,
       data,dim);
12   }while( status!=HAL_OK && try <=
       max_retry );
13
14   return status;
15
16 }
17
```

Inoltre scritte funzioni di libreria per la gestione della interrupt mode.

Interfaccia audio

Conversione timer driven ADC 1/2 (mic1/mic2) e DAC (ch1/ch2) su timer 2 update event.
Configurazioni:

1. Frequenza di campionamento:

$$F_{tim2} = F_{clock}/(prescaler{+}1){\cdot}(period{+}1) = 25{\cdot}10^6/2{\cdot}256 = 48.828kHz$$

.
2. DMA continous request flag abilitato.
3. DMA in circular mode.
4. Continous conversion mode disabilitata - è la TRGO line del timer che comanda la conversione.

Input pin: PB0 e PB7.
Ouput pin: PA4 e PA5.

Motore stepper

Quattro pin di output collegati al chip driver. Attivare in pin secondo un determinato driving pattern per muovere il motore in un senso o nell'altro. Di seguito mostrate wave e full drive.

pin0	pin1	pin2	pin3
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

pin0	pin1	pin2	pin3
1	1	0	0
0	1	1	0
0	0	1	1
1	0	0	1

Driver motore

```c
void waveStep(Step *inst,uint8_t dir){

    /*Reset previous step pin*/
    inst->port->ODR&= ~(inst->pins[ inst->
      cur_step ]);

    if(dir){

        /*Move forward cur_step*/
        inst->cur_step=(inst->cur_step +
      1) & 0x3;
        inst->ang_idx ++;
    }
    else{

        /*Move backward cur_step*/
        inst->cur_step= inst->cur_step ? (
      inst->cur_step - 1) : 0x3;
        inst->ang_idx --;
    }

    /*Set current step pin*/
    inst->port->ODR|= inst->pins[ inst->
      cur_step ];
}
```

```c
void fullStep(Step *inst,uint8_t dir){
    if(dir){
        /*Reset previous step pin*/
        inst->port->ODR&= ~(inst->pins[
      inst->cur_step ]);

        /*Move forward cur_step*/
        inst->cur_step=(inst->cur_step +
      1) & 0x3;
        inst->ang_idx ++;
    }
    else{
        /*Reset previous step right
      sibling pin*/
        inst->port->ODR&= ~(inst->pins[ (
      inst->cur_step + 1) & 0x3 ]);
        /*Move backward cur_step*/
        inst->cur_step= inst->cur_step ? (
      inst->cur_step - 1) : 0x3;
        inst->ang_idx --;
    }
    /*Set current step pin and its right
      sibling*/
    inst->port->ODR|= ( inst->pins[ inst->
      cur_step ] | inst->pins[ (inst->
      cur_step + 1) & 0x3] );
```

Motore stepper

```
/*
 * Move motor from a starting position to
      a destination expressed in angle
      degrees in interrupt mode
 */
void moveToIt(Step *inst,float angle){

    if(!inst->move_lock){
        inst->move_lock=1;

        inst->destination_it=angle;
        HAL_TIM_Base_Start_IT( inst->htim
      );
    }
}


```

```
/*
 * Perform one step in interrupt mode
 * this function is meant to be called
      inside instance timer PeriodElapsed
      callback
 */
void stepIt(Step *inst){
    if( inst->destination_it > inst->
      ang_idx * inst->res ){

        if( inst->destination_it > ( inst
      ->ang_idx + 1 ) * inst->res )
            step(inst,1);
        else{
            HAL_TIM_Base_Stop_IT(inst->
      htim);
            inst->move_lock=0;
        }
    }

    else{
        if( inst->destination_it < ( inst
      ->ang_idx - 1 ) * inst->res )
            step(inst,0);
        else{
            HAL_TIM_Base_Stop_IT(inst->
      htim);
            inst->move_lock=0;
        }
    }
}
```

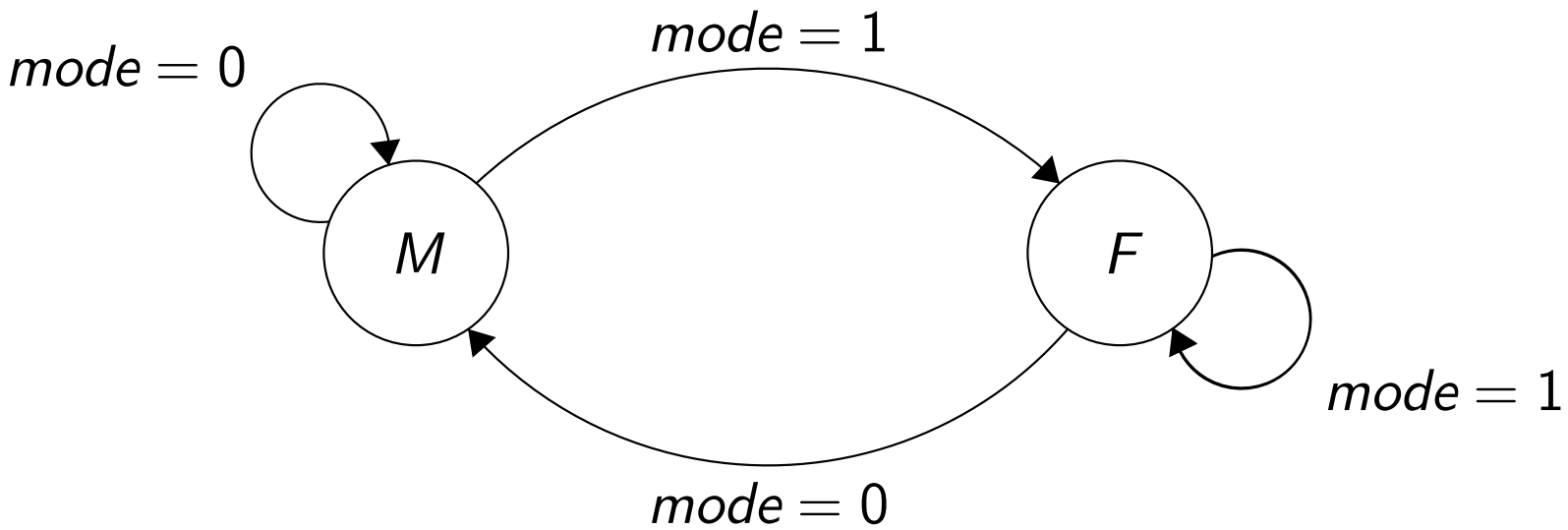
Joystick, exti lines e UART

1. Joystick: alimentato a 3V, conversione in polling mode del valore di tensione in ingresso all'ADC3. Single channel/conversion mode. Lettura mono-asse, bidirezionale (destra/sinistra).
2. Bottoni di controllo: switch modalità, controllo motore singolo step. Input pin con internal pull-up, external interrupt falling edge mode. La pressione di un tasto attiva un timer di debounce allo scadere del quale lo stato dei pin d'ingresso viene ricontrollato.
3. AMG8833 INT pin: external interrupt falling edge mode, resistenza di pull-up esterna (ingresso open drain).
4. UART3 per output immagine termica e UART6 per print di debug.

Pattern architetturale

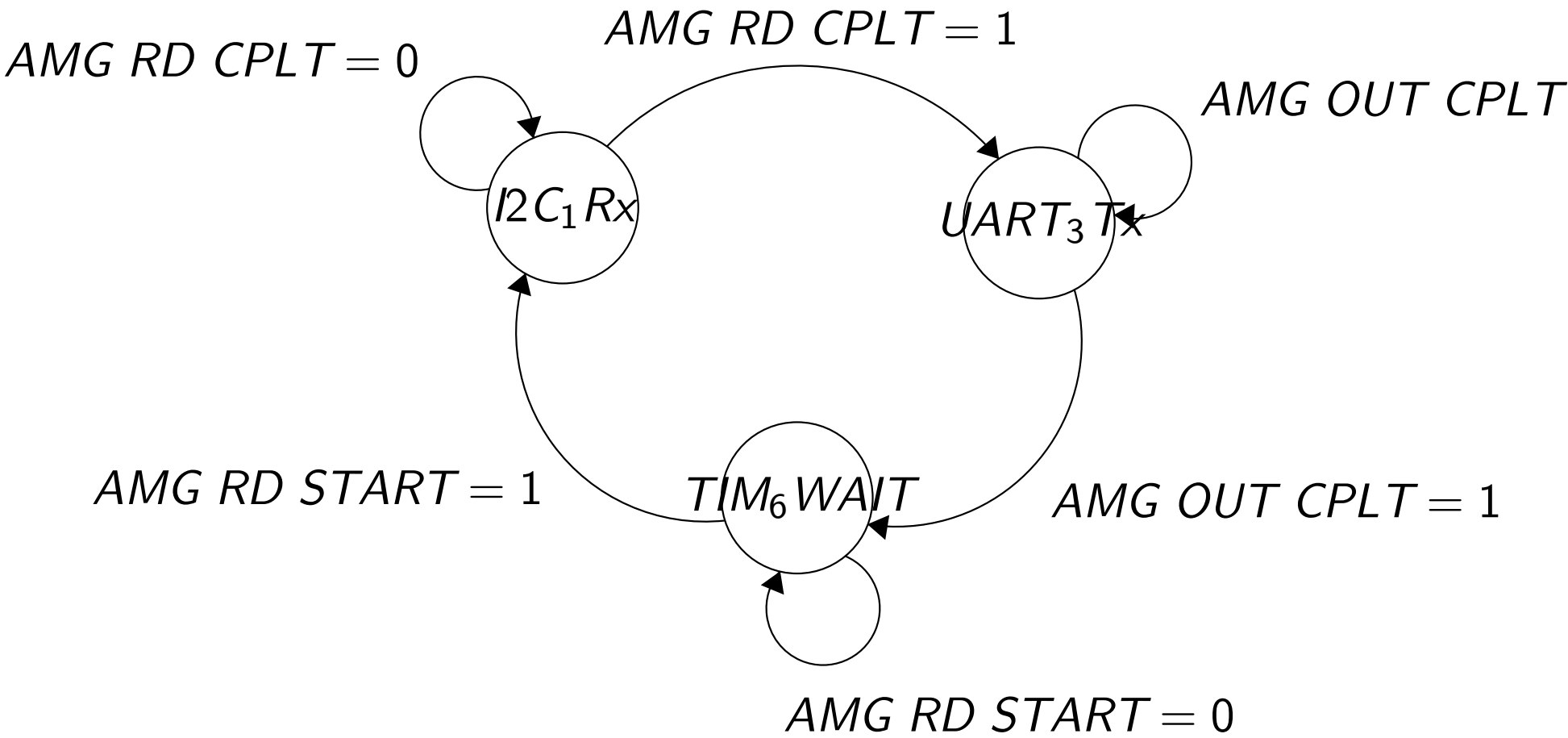
Difficoltà principale del progetto: controllo simultaneo di diverse periferiche rispettando le deadline imposte dalla conversione audio, dal frame rate di acquisizione dell'immagine termica e dal movimento fluido del motore. Impossibile gestire tutte le periferiche in polling sequenziale.
Obiettivi: architettura uniforme e comprensibile. Codice ISR efficiente e non-error-prone.
Soluzione: set/reset di status flag in bit band region nelle ISR e polling occasionale su questi ultimi nel loop di controllo.

Modalità



Switch tra modalità manuale e modalità follow. Flag mode settato da exti button.

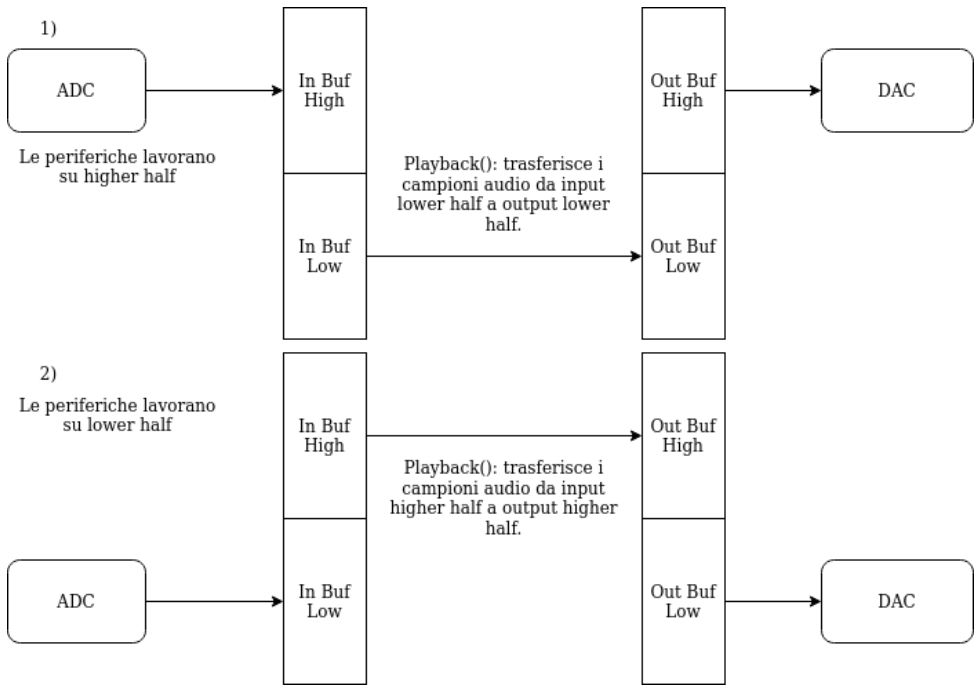
Thermal image FSM



```
1  void thermalImgFSM(){
2      //If timer6 has expired
3      if(AMG_RD_START){
4          //Command DMA transfer from
       amg8833
5          status=amg8833ReadDMA( &cam,
       img_buf, 1 );
6          if(status==HAL_OK){
7              //Clear ctrl read start bit
8              AMG_RD_START=0;
9          }
10     }
11     //If DMA image reading was successful
12     if(AMG_RD_CPLT){
13         //Command DMA transfer to uart2
14         status=HAL_UART_Transmit_DMA(&
       huart3,img_buf,AMG8833_DS);
15         if(status==HAL_OK){
16             AMG_RD_CPLT=0;
17         }
18     }
19     //if latest data were consumed in
       output, restart timer6
20     if(AMG_OUT_CPLT){
21         AMG_OUT_CPLT=0;
22         HAL_TIM_Base_Start_IT(&htim6);
23     }
24 }
25
```

```
1  void HAL_I2C_MemRxCpltCallback(
       I2C_HandleTypeDef *hi2c){
2      if(hi2c->Instance == I2C1){
3          AMG_RD_CPLT=1;
4      }
5  }
6
7  void HAL_UART_TxCpltCallback(
       UART_HandleTypeDef *huart){
8      if(huart->Instance == USART3){
9          AMG_OUT_CPLT=1;
10     }
11 }
12 void HAL_TIM_PeriodElapsedCallback(
       TIM_HandleTypeDef *htim)
13 {
14     //...
15     else if( htim->Instance == TIM6 ){
16         AMG_RD_START=1;
17     }
18     //...
19 }
```

Audio playback: ping-pong buffers



Taglia totale di buffer $=$ 1024 samples $\Longrightarrow$ half buffer $=$ 512 (0.01sec).

Modalità manuale

```
1  void audioPlayback(){
2      uint32_t mean1,mean2;
3      /*If channel 1 and 2 conversion was
         completed*/
4      if(BUF1_CPLT && BUF2_CPLT){
5          BUF1_CPLT=0;
6          BUF2_CPLT=0;
7
8          /*Transfer samples from input to
         output buffers*/
9          for(int i=0;i<AUDIO_BUF_SZ;i++){
10             /*
11             * Compute mean value in
         buffers 1 and 2
12             */
13             mean1+=audio_in_ptr1[i];
14             mean2+=audio_in_ptr2[i];
15
16             audio_out_ptr1[i]=
         audio_in_ptr1[i];
17             audio_out_ptr2[i]=
         audio_in_ptr2[i];
18         }
19     }
20 }
21
```

```
1  void HAL_ADC_ConvHalfCpltCallback(
         ADC_HandleTypeDef *hadc){
2      if(hadc->Instance==ADC1){
3          audio_in_ptr1=&audio_in_buf1[0];
4          audio_out_ptr1=&audio_out_buf1[0];
5          BUF1_CPLT=1;
6      }
7      if(hadc->Instance==ADC2){
8          audio_in_ptr2=&audio_in_buf2[0];
9          audio_out_ptr2=&audio_out_buf2[0];
10         BUF2_CPLT=1;
11     }
12 }
13 void HAL_DAC_ConvCpltCallbackCh1(
         DAC_HandleTypeDef *hdac){
14     audio_in_ptr1=&audio_in_buf1[
         AUDIO_BUF_SZ];
15     audio_out_ptr1=&audio_out_buf1[
         AUDIO_BUF_SZ];
16     BUF1_CPLT=1;
17 }
18 void HAL_DACEx_ConvCpltCallbackCh2(
         DAC_HandleTypeDef *hdac){
19     audio_in_ptr2=&audio_in_buf2[
         AUDIO_BUF_SZ];
20     audio_out_ptr2=&audio_out_buf2[
         AUDIO_BUF_SZ];
21     BUF2_CPLT=1;
22 }
```

Controllo motore

```
1  void motorControl(){
2      JstickDir dir;
3
4      if(MOTOR_MV){
5          MOTOR_MV=0;
6
7          dir=jstickGetDirPoll(&js);
8          if(dir==LEFT || LEFT_BUT_PUSH){
9              LEFT_BUT_PUSH=0;
10             step(&motor,1);
11         }
12         else if(dir==RIGHT || RIGHT_BUT_PUSH){
13             RIGHT_BUT_PUSH=0;
14             step(&motor,0);
15         }
16     }
17
18 }
```

Il flag MV è settato dal timer 7 che ogni 2 ms triggera una lettura dei controlli manuali e il conseguente movimento del motore.

Event trigger

Nella modalità follow (o SSL) il dispositivo reagisce a stimoli sonori nell'ambiente circostante e cerca di identificare il target che ha prodotto il rumore, per orientare verso di esso la termocamera.
Pertanto, viene monitorata la potenza RMS di ogni buffer audio in attesa di eventi sopra una soglia preimpostata. Per calcolare RMS è necessario che il segnale sia a media 0: bisogna rimuovere il DC offset.
Il DC offset, dipendente dalla precisione del circuito di preamplificazione, viene stimato calcolando la media degli ultimi 100 valori medi dei buffer.

Cross-correlazione

Il metodo classico per stimare l'angolo d'incidenza di una sorgente sonora è la cross-correlazione dei segnali. L'operatore permette di calcolare la Time Difference Of Arrival dello stesso segnale sui due microfoni.
Probelemi:

1. Fattori di imprecisione nel circuito di condizionamento costruito ad-hoc per il progetto.
2. Riflessioni ambientali.

L'implementazione è risultata inefficace e molto difficile da debuggare, poichè a livello teorico l'algoritmo assume un ambiente privo di riflessioni ed estrema precisione nell'acquisizione (i segnali devono essere identici a meno di amplificazione e sfasamento).

Workaround e proof-of-concept

Per evitare una lunga attività di debug e riprogettazione, si è pensato di implementare una proof-of-concept della modalità follow per mostrare il processo di automazione sfruttando le feature già implementate.
Alla rilevazione di almeno due buffer sopra soglia il dispositivo:

1. Abilità la interrupt mode in absolute value sul dispositivo AMG8833.
2. Esegue una pan su tutto il campo visivo in interrupt mode, senza interrompere playback audio e streaming video.
3. Il motore viene arrestato quando nell'ottica visuale viene rilevato un oggetto sopra la soglia di temperatura impostata: il sensore genera un interrupt sul pin INT.

Tempi di esecuzione e consumi

Per verificare il corretto funzionamento è necessario misurare:

1. Frame rate streaming video. Un ciclo completo di I/O ogni 25 ms. Ogni frame è ridondato 4 volte. Pro: sicurezza di non perdere frame, tecnica compatibile con un dispositivo con frame rate maggiore. Contro: maggior dispendio di energia.
2. Conversione audio/ DMA callback. 10 ms.
3. Tempo impiegato da loop di playback e preprocess per portare campioni da input buffer ad ouput buffer e calcolare dc offset e rms. 1 ms.

Tecnica utilizzata: approccio hardware con oscilloscopio.
Ragioni: disponibilità di strumento e uscita digitale. Rimane l'approccio che garantisce maggiore precisione.
In entrambe le modalità il dispositivo consuma 20mAh.

Sviluppi futuri

1. Miglioramento elaborazione dei segnali.
2. Ottimizzazione consumi energetici.
3. Remote control.

DEMO