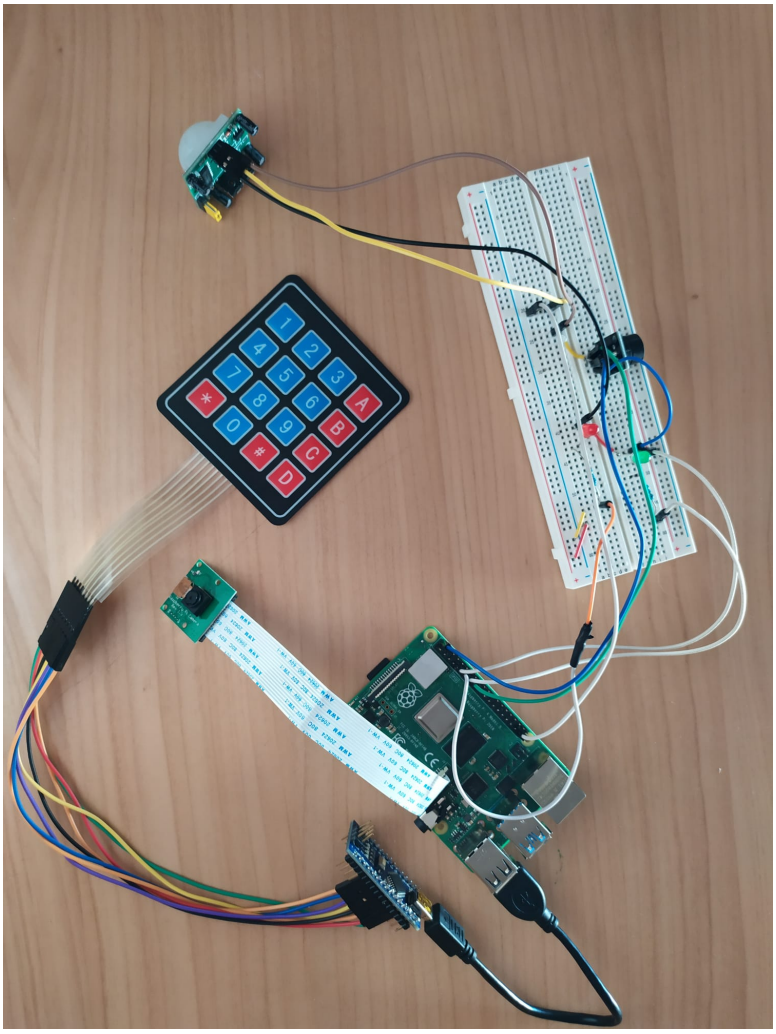
Un implementazione di un sistema antifurto con Arduino e Raspberry Pi

Lorenzo Mustich

22 luglio 2020



Indice

1 Introduzione

Il progetto in questione tratta di un'antifurto da appartamento che utilizza un Raspberry Pi4 come unità centrale di elaborazione, un Arduino come controllo dell'input da utente e due bot Telegram per l'attivazione e disattivazione e per l'invio dei video. L'antifurto può essere attivato e disattivato in due modi:

- utilizzando un bot scritto appositamente: **AntiTheftOnOff** (***@ATOnOffBot***);
- digitando una serie di caratteri alfanumerici per mezzo del tastierino a membrana, nel caso in cui non si abbia la suddetta applicazione.

Una volta attivato, un sensore PIR, appositamente tarato, rileverà eventuali movimenti all'interno del suo campo visivo innescando l'accensione di una spia e di un allarme prodotto da un buzzer attivo; successivamente, una videocamera produrrà un filmato di pochi secondi dell'area. Un secondo bot Telegram (**ATVideoBot**, ***@ATVideoBot***) è adibito all'invio del video al proprietario della casa.

2 Componenti

Di seguito è mostrata la lista dei componenti:

- Raspberry Pi4;
- Arduino;
- Sensori:
  - HC-SR501, sensore PIR di movimento;
  - Buzzer attivo;
  - PiCamera;
  - Tastiera a membrana 4x4 16 tasti;
- Led rosso/verde;
- Due resistenze da 220 ohm

Figura 1: Componenti del sistema

2.1 HC-SR501 Sensore PIR

Si tratta di un sensore composto da due slot di materiale sensibile agli infrarossi. Quando è in stato di *idle*, i due slot captano la stessa quantità di raggi; nel momento in cui un corpo caldo entra nella zona d'azione del sensore intercetterà il primo dei due slot creando una differenza di potenziale all'interno

del sensore. Quando il corpo lascia il campo visivo, esso intercetterà il secondo slot portando la differenza di potenziale ad un valore negativo. Il sensore è in grado di rilevare questo cambiamento. Il **sensore HC-SR501** ha un raggio d'azione di 110 gradi conici con una distanza che va dai 3 ai 7 metri.È possibile impostare per quanto tempo l'output del PIR deve essere tenuto alto dopo la rilevazione del movimento: questo intervallo va dai 3 secondi ai 5 minuti.

3 Architettura

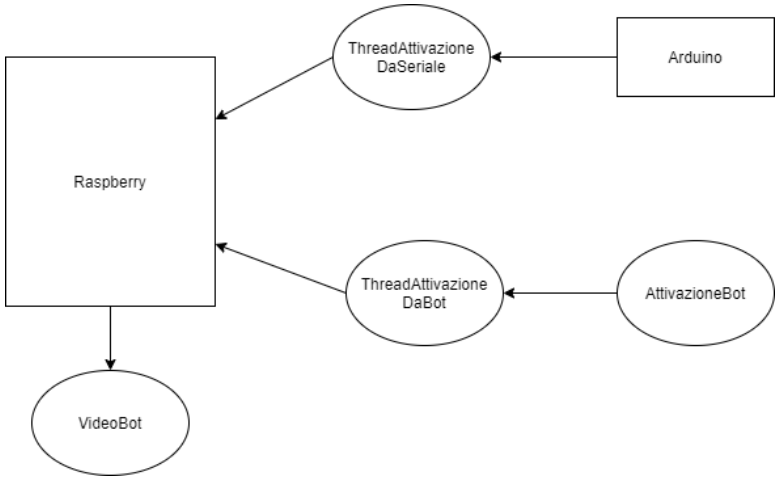


Figura 2: Architettura del sistema

Raspberry ed Arduino comunicano tramite un cavo USB: alla pressione dei tasti corrispondenti alla password viene scritto un carattere su seriale che viene recepito dal Raspberry, avviando l'antifurto. Come spiegato precedentemente, l'attivazione può avvenire in due modi, utilizzando un bot apposito o il tastierino a disposizione. Questi due metodi sono implementati da due Thread che si occupano indipendentemente delle due modalità:

- nel primo caso, il microcontrollore confronta l'input da utente con la password scelta;
- nel secondo caso, l'utente invia dei comandi di attivazione e disattivazione tramite bot.

Per notificare l'avvenuta accensione e spegnimento del sistema è stato posto un led rosso lampeggiante.

4 Circuito

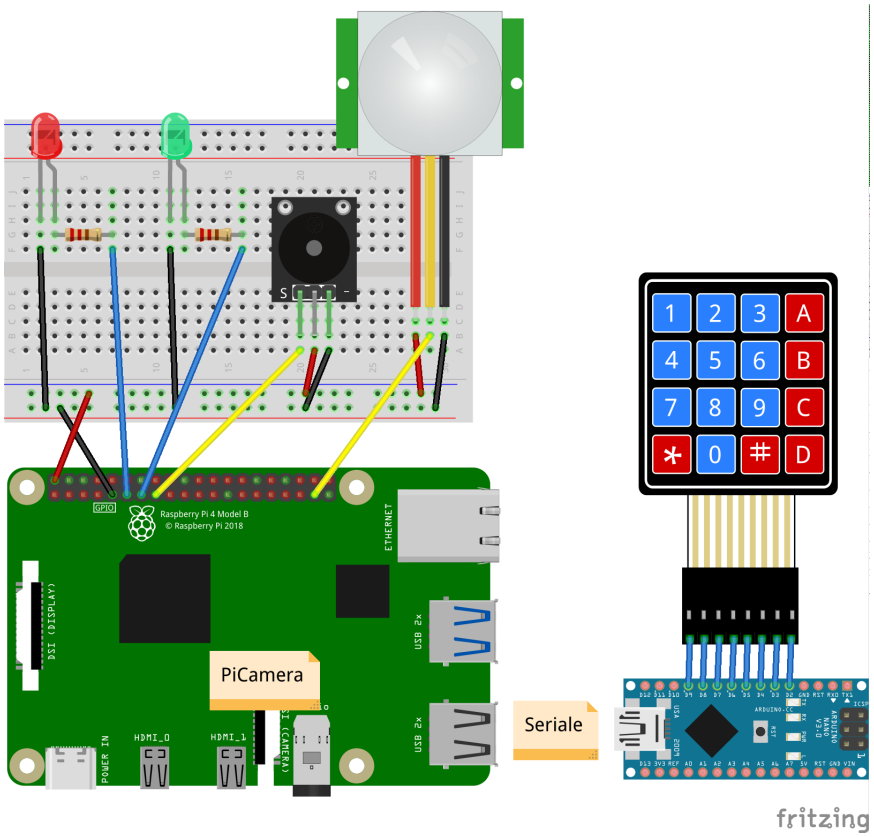


Figura 3: Circuito

5 Librerie

Ai fini del corretto funzionamento del sistema è necessario avere a disposizione le seguenti librerie:

- Python (Raspberry Pi):
  - modulo per bot Telegram: *pip3 install python-telegram-bot*;
  - modulo per la gestione della camera: *pip3 install picamera*;
  - modulo per la lettura da seriale: *pip3 install pyserial*;
  - modulo per la conversione dei video: *pip3 install ffmpeg-python*;
- Arduino:
  - libreria Arduino per la gestione della tastiera: *"Keypad.h"*

6 Codice

Di seguito è presentato lo script principale del sistema: *antifurto.py*

```
import os
import telegram

from picamera import PiCamera
from time import sleep

import RPi.GPIO as GPIO
import serial

from threading import Thread, Lock
import ffmpeg

#onoff.py
import onoff

TOKEN = "YOUR_TOKEN"

bot = telegram.Bot(TOKEN)

camera = PiCamera()

#Initialize Raspberry's pinMode and pins
GPIO.setmode(GPIO.BCM)
GPIO.setup(26, GPIO.IN)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(27, GPIO.OUT)
GPIO.setup(22, GPIO.OUT)

#Initialize Raspberry serial
ser = serial.Serial("/dev/ttyUSB0", 9600)

#Initialize variables for threads
activated = False
numState = 0
mutex = Lock()

#Activation by serial Thread
class activateSerialThread(Thread):
    def run(self):
        global activated, numState

        while True:
            if ser.read().decode('ascii') == 'y':
                mutex.acquire()

                if numState % 2 == 0:
```

```
                print("ATTIVATO SERIALE!")
                activated = True
            else:
                print("DISATTIVATO SERIALE!")
                activated = False

            numState = numState + 1

            for i in range(3):
                GPIO.output(17, GPIO.HIGH)
                sleep(0.1)
                GPIO.output(17, GPIO.LOW)
                sleep(0.1)

            mutex.release()


#Activation by bot Thread
class activateBotThread(Thread):
    def run(self):
        global activated, numState

        while True:
            if onoff.telegram_act is not None:
                if onoff.telegram_act:
                    mutex.acquire()

                    print("ATTIVATO!")
                    activated = True
                    onoff.telegram_act = None

                    numState = numState + 1

                elif not onoff.telegram_act:
                    mutex.acquire()

                    print("DISATTIVATO!")
                    activated = False
                    onoff.telegram_act = None

                    numState = numState + 1

                for i in range(3):
                    GPIO.output(17, GPIO.HIGH)
                    sleep(0.1)
                    GPIO.output(17, GPIO.LOW)
                    sleep(0.1)

                mutex.release()
```

```
#Initialize PIR variables
pirState = GPIO.LOW
val = 0

activateST = activateSerialThread()
activateBT = activateBotThread()

activateST.start()
activateBT.start()

onoff.update_bot()


while True:
    if activated:
        val = GPIO.input(26)

        if val == GPIO.HIGH:
            if pirState == GPIO.LOW:
                print("Motion detected\n")
                pirState = GPIO.HIGH

                GPIO.output(27, GPIO.HIGH)
                GPIO.output(22, GPIO.HIGH)

                #PiCamera records only in .mjpeg or .
                   h264
                camera.start_recording("/home/pi/video.
                   h264")
                sleep(10)
                camera.stop_recording()

                if bot.get_updates():
                    chat_id = bot.get_updates()[-1].
                       message.chat_id

                    #The version of Telegram for
                       smartphone is unable to read file
                        .mjpeg or .h264.
          In order to achieve this, it is important to
             convert the video file in .mp4
                    os.system("ffmpeg -y -r 15 -i /home/
                       pi/video.h264 -an -c:v copy /home
                       /pi/video.mp4 > /dev/null 2>&1")
                    bot.send_video(chat_id, video = open
                       ("/home/pi/video.mp4", 'rb'),
                       supports_streaming = True)
```

```
                else:
                    print("Errore")

        else:
            GPIO.output(27, GPIO.LOW)
            GPIO.output(22, GPIO.LOW)

            if pirState == GPIO.HIGH:
                print("Motion ended\n")
                pirState = GPIO.LOW

else:
    GPIO.output(27, GPIO.LOW)
    GPIO.output(22, GPIO.LOW)

    if pirState == GPIO.HIGH:
        pirState = GPIO.LOW
```

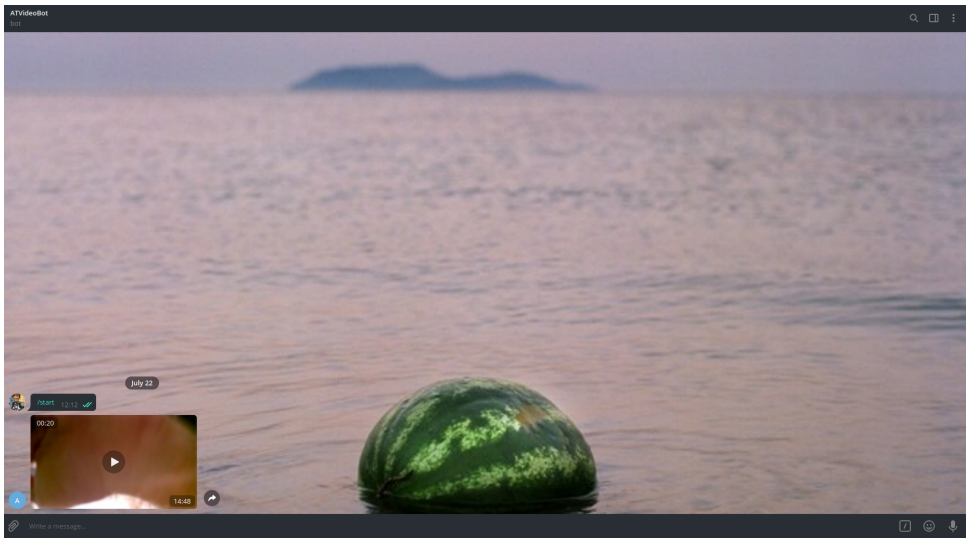


Figura 4: Bot per ricezione video

Di seguito il codice del bot di attivazione: *onoff.py*

```
from telegram.ext import Updater, CommandHandler

TOKEN_ONOFF = "YOUR_TOKEN"

telegram_act = None

#Start the bot
def start(update, context):
    update.message.reply_text("/attiva: attiva l\'
       antifurto\n/disattiva: disattiva l\'antifurto\n/
       help: visualizza i comandi")

#Print available commands
def help_command(update, context):
    update.message.reply_text("/attiva: attiva l\'
       antifurto\n/disattiva: disattiva l\'antifurto\n/
       help: visualizza i comandi")

#Turn on the system
def attiva(update, context):
    global telegram_act

    update.message.reply_text("Antifurto attivato!")
    telegram_act = True

#Turn off the system
def disattiva(update, context):
    global telegram_act

    update.message.reply_text("Antifurto disattivato!")
    telegram_act = False


def update_bot():
    updater = Updater(TOKEN_ONOFF, use_context=True)

    updater.dispatcher.add_handler(CommandHandler('start
       ', start))
    updater.dispatcher.add_handler(CommandHandler('help'
       , help_command))
    updater.dispatcher.add_handler(CommandHandler('
       attiva', attiva))
    updater.dispatcher.add_handler(CommandHandler('
       disattiva', disattiva))

    updater.start_polling()
```

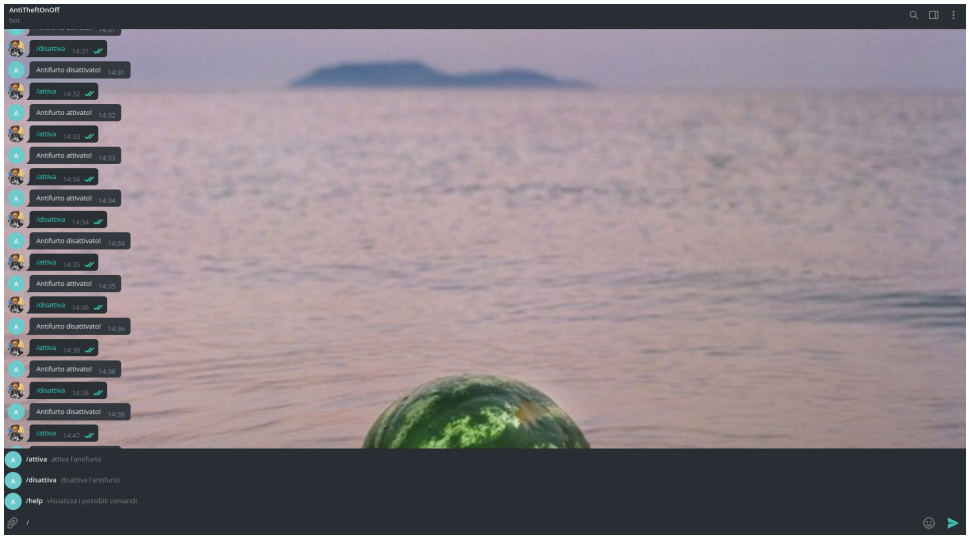


Figura 5: Bot per attivazione e disattivazione antifurto

Di seguito il codice relativo ad Arduino: *check_password.ino*

```
#include <Keypad.h>

const byte ROWS = 4;
const byte COLS = 4;

const byte PASSWD_LEN = 5;
char passwd[PASSWD_LEN] = "YOUR_PASSWORD";
char data[PASSWD_LEN];

int counter = 0;

char hexaKeys[ROWS][COLS] = {
    {'1', '2', '3', 'A'},
    {'4', '5', '6', 'B'},
    {'7', '8', '9', 'C'},
    {'*', '0', '#', 'D'}
};

byte pinRows[ROWS] = {9, 8, 7, 6};
byte pinCols[COLS] = {5, 4, 3, 2};

Keypad customKeypad = Keypad(makeKeymap(hexaKeys),
   pinRows, pinCols, ROWS, COLS);


void setup() {
    Serial.begin(9600);
}
```

```
void loop() {
    char customKey = customKeypad.getKey();

    if (customKey) {
      data[counter] = customKey;
      counter++;

      if (counter == PASSWD_LEN - 1) {
          data[counter] = '\0';

        if (!strcmp(data, passwd))
          Serial.print("y");

      counter = 0;
      }//counter
    }//customkey

}
```

Per permettere al sistema antifurto di essere attivo all'accensione del Raspberry è possibile inserire nel file *.bashrc* la seguente linea di codice:

```
python3 ./antifurto.py
```

7 Conversione video

La PiCamera produce video in soli due formati, .h264 e .mjpeg, che non permettono la loro fruizione da Telegram nella sua versione per smartphone. È stato necessario introdurre un passaggio di conversione nel codice che permettese l'invio di file.mp4.

ffmpeg -y -r 15 -i /home/pi/video.h264 -an -c:v copy /home/pi/video.mp4 > /dev/null > 2&1

Di seguito la spiegazione dei vari parametri:

- -y: sovrascrive i file di output senza chiedere all'utente;
- -r: numero di fps;
- -i: percorso del file di input;
- -an: esclude dal flusso l'audio;
- -c:v copy: non effettua un nuovo encoding sul flusso video in uscita;
- -o: percorso del file di output

8 Conclusioni

Il sistema presentato è solo un prototipo: un possibile sviluppo futuro potrà essere quello di implementare un prodotto finito con circuiti stampati e involucro esterno. Inoltre, non è presente alcunchè che possa essere utile all'utente in caso di errori: potrebbe, perciò, essere comodo inserire un qualsiasi dispostivo dotato di schermo per facilitare il debug in loco.