Università degli Studi di Salerno

Corso di Laurea in Ingegneria Informatica

Documentazione Progetto IoT

Annamaria Scermino, Anuar Zouhri, Gerardo Selce

TheBox

2 giugno 2025

Indice

1 Introduzione

Il progetto sviluppato ha come obiettivo la realizzazione di uno smart caveaux. Il caveaux è dotato di un meccanismo di apertura che può essere controllato sia tramite un tastierino fisico posto all'esterno, sia da remoto attraverso una piattaforma di controllo. Il sistema integra inoltre sensori per la rilevazione di temperatura e umidità, con la capacità di attivare allarmi in caso di valori fuori soglia o tentativi di intrusione. Gli allarmi possono essere gestiti localmente o a distanza. Un semaforo posto accanto alla porta mostra lo stato del caveaux.

2 Architettura del Sistema

I componenti hardware utilizzati sono i seguenti:

- Un servomotore;
- Un tastierino numerico;
- Uno schermo OLED;
- Due breadboard;
- Due push-button;
- Un semaforo;
- Un sensore ad ultrasuoni;
- Un sensore di umidità e temperatura;
- Un buzzer passivo;
- Una ESP32
- Un relay

Dal punto di vista software, il sistema è stato sviluppato interamente in micropython. Per il controllo remoto e il monitoraggio in tempo reale è stata realizzata una dashboard interattiva con Node-RED, che consente di visualizzare lo stato del sistema, ricevere notifiche e inviare comandi. Inoltre alcune funzionalità, come l'apertura del caveaux, sono accessibili anche da un'app mobile. I protocolli utilizzati sono: MQTT ed I2C.

3 Descrizione del Codice

Sono state riportate solo le classi ritenute particolarmente significative.

3.1 File: main.py

```
while True:

        if stato == STATO_CONFIGURAZIONE_PIN:
                pass
                #
                # AGGIORNAMENTO DELLO STATO
                #
                stato = NUOVO_STATO

        elif stato == STATO_CONFIGURAZIONE_WIFI:
                pass
                #
                # AGGIORNAMENTO DELLO STATO
                #
                stato = NUOVO_STATO

        elif stato == STATO_CONNESSIONE:
                pass
                #
                # AGGIORNAMENTO DELLO STATO
                #
                stato = NUOVO_STATO

        elif stato == STATO_VISTA_MENU:
                pass
                #
                # AGGIORNAMENTO DELLO STATO
                #
                stato = NUOVO_STATO

        elif stato == STATO_CAMBIO_CONFIGURAZIONE:
                pass
                #
                # AGGIORNAMENTO DELLO STATO
                #
                stato = NUOVO_STATO

        elif stato == STATO_INSERIMENTO_PIN:
                pass
                #
                # AGGIORNAMENTO DELLO STATO
                #
                stato = NUOVO_STATO

        elif stato == STATO_SBLOCCATO:
                pass
                #
                # AGGIORNAMENTO DELLO STATO
                #
```

```
50                  stato = NUOVO_STATO
51
52          elif stato == STATO_ALLARME:
53                  pass
54                  #
55                  # AGGIORNAMENTO DELLO STATO
56                  #
57                  stato = NUOVO_STATO
```

Listing 1: Il main del programma è stato sviluppato come un automa a stati a finiti e qui sono rappresentati i vari stati. Lo stato di configurazione pin è lo stato che viene eseguito alla prima esecuzione del programma: qui viene configurato il file contenente il pin d'accesso al caveaux. Lo stato di configurazione wifi è eseguito sequenzialmente a quello di configurazione pin. Lo stato di connessione permette la connessione all'MQTT. Finiti i vari step di connessione si accede allo stato di vista menu, in cui vengono riportati parametri interni al caveaux, quali temperatura e umidità, ed è inoltre possibile scegliere di accedere al caveaux o di cambiare configurazione. Entrambe le opzioni appena descritte vengono eseguite in appositi stati: stato di inserimento pin e stato di cambio configurazione. Una volta sbloccato il caveaux si entra nello stato sbloccato. In caso di intrusione o valori di temperatura e umidità fuori norma si entra nello stato di allarme.

3.2 File: Handler.py

```
1  class SensorHandler:
2
3          """ Costruttore """
4          def __init__(self,dht22, hcsr04, sogliaTemp=50.0, sogliaHum
               =70.0, sogliaDis=10.0):
5                  self.dht22 = dht22
6                  self.hcsr04 = hcsr04
7                  self.sogliaTemp = 0.0
8                  self.sogliaHum = 0.0
9                  self.sogliaDis = 0.0
10
11                 if 'temp.txt' in os.listdir():
12                         with open('temp.txt', 'r') as f:
13                                 self.sogliaTemp = float(f.read())
14                 else:
15                         with open('temp.txt', 'w') as f:
16                                 f.write(str(sogliaTemp))
17                                 self.sogliaTemp = sogliaTemp
18
19                 if 'hum.txt' in os.listdir():
20                         with open('hum.txt', 'r') as f:
21                                 self.sogliaHum = float(f.read())
22                 else:
23                         with open('hum.txt', 'w') as f:
24                                 f.write(str(sogliaHum))
25                                 self.sogliaHum = sogliaHum
26
```

```
27                  if 'dis.txt' in os.listdir():
28                          with open('dis.txt', 'r') as f:
29                                  self.sogliaDis = float(f.read())
30                  else:
31                          with open('dis.txt', 'w') as f:
32                                  f.write(str(sogliaDis))
33                                  self.sogliaDis = sogliaDis
34
35
36
37          """ Legge i valori e ne crea un dizionario """
38          def readDht22(self):
39                  dht_values = self.dht22.measure()
40                  dict_values = { "Temperature": dht_values[0] ,
41                                                  "Humidity":
                                                        dht_values[1]
42                                          }
43                  return dict_values
44
45
46          def readHcsr04(self):
47                  return self.hcsr04.distanceCm()
48
49
50          def checkDistance(self):
51                  return self.readHcsr04() < self.sogliaDis
52
53
54          def checkTempHum(self, temp, hum):
55                  if temp is not None and temp >= self.sogliaTemp:
56                          return 1
57                  elif hum is not None and hum >= self.sogliaHum:
58                          return 2
59
60                  return 0
61
62          def setSogliaTemp(self, sogliaTemp):
63                  self.sogliaTemp = sogliaTemp
64                  with open('temp.txt', 'w') as f:
65                          f.write(str(sogliaTemp))
66
67          def setSogliaHum(self, sogliaHum):
68                  self.sogliaHum = sogliaHum
69                  with open('hum.txt', 'w') as f:
70                          f.write(str(sogliaHum))
71
72          def setSogliaDis(self, sogliaDis):
73                  self.sogliaDist = sogliaDist
74                  with open('dis.txt', 'w') as f:
75                          f.write(str(sogliaDis))
```

Listing 2: Questa classe gestisce la lettura del sensore ad ultrasuoni (hcsr04) e del sensore di umidità e temperatura (dht22) e il controllo che i valori letti siano minori di una certa soglia. Il costruttore inizializza le soglie con i valori imposti alla precedente esecuzione (leggendo da file) o tramite parametri di input. I metodi read() e check() permettono rispettivamente la lettura dei valori e il loro controllo. È inoltre possibile modificare le soglie tramite metodi set().

3.3 File: `MQTTClass.py`

```
class MQTT:

        """ Costruttore """
        def __init__(self, sub_callback_handler):
                """ Definizione variabili connessione MQTT """
                self.CLIENT_ID   = 'esp32'
                self.BROKER      = '192.168.131.51'
                self.USER        = ''
                self.PASSWORD    = ''
                self.TOPIC       = b'TheBox'
                self.TOPIC_SUB1  = b'TheBox/OpenCaveaux'
                self.TOPIC_SUB2  = b'TheBox/CloseCaveaux'
                self.TOPIC_SUB3  = b'TheBox/CaveauxStatus'
                self.TOPIC_SUB4  = b''
                self.TOPIC_SUB5  = b'TheBox/CaveauxStatus/Hum'
                self.TOPIC_SUB6  = b'TheBox/Pin'
                self.TOPIC_SUB7  = b'TheBox/Pin/WrongPassword'
                self.TOPIC_SUB8  = b'TheBox/Allarme/Furto'
                self.TOPIC_SUB9  = b'TheBox/Allarme/Risolto'
                self.TOPIC_SUB10 = b'TheBox/Allarme/PinErrato'
                self.TOPIC_SUB11 = b'TheBox/Status/Ping'
                self.TOPIC_SUB12 = b'TheBox/Status/Pong'
                self.TOPIC_SUB13 = b'TheBox/Allarme/StopBuzzer'
                self.TOPIC_SUB14 = b'TheBox/CaveauxStatus/Temp/
                   NuovaSoglia'
                self.TOPIC_SUB15 = b'TheBox/CaveauxStatus/Hum/
                   NuovaSoglia'
                self.TOPIC_SUB16 = b'TheBox/CaveauxStatus/Dis/
                   NuovaSoglia'
                self.SUB_TOPICS  = [self.TOPIC_SUB1,
                self.TOPIC_SUB2, self.TOPIC_SUB3,
                self.TOPIC_SUB4, self.TOPIC_SUB5,
                self.TOPIC_SUB6, self.TOPIC_SUB7,
                self.TOPIC_SUB8, self.TOPIC_SUB9,
                self.TOPIC_SUB10, self.TOPIC_SUB11,
                self.TOPIC_SUB12, self.TOPIC_SUB13,
                self.TOPIC_SUB14, self.TOPIC_SUB15,
                self.TOPIC_SUB16]

                self.SUB_TOPICS_SUB=[self.TOPIC_SUB2,
```

```
38                  self.TOPIC_SUB6, self.TOPIC_SUB11,
39                  self.TOPIC_SUB13, self.TOPIC_SUB14,
40                  self.TOPIC_SUB15, self.TOPIC_SUB16]
41
42                  """ Inizializzazione dell'oggetto client """
43                  self.client = MQTTClient(self.CLIENT_ID,
44                  self.BROKER, user=self.USER,
45                  password=self.PASSWORD, keepalive=60)
46                  self.sub_callback_handler = sub_callback_handler
47                  self.client.set_callback(self.sub_callback_handler)
48
49                  """ Inizializzazione oggetto MMaker """
50                  self.mMaker = MM()
51
52
53          """ Il client si sottoscrive ai topics """
54          def subscribes(self):
55                  for topic in self.SUB_TOPICS_SUB:
56                          self.client.subscribe(topic)
57
58
59          def checkAndRead_msg(self, wifi, was_connected_MQTT, values
                ):
60                  if wifi.isconnected() and self.client is not None
                        and was_connected_MQTT:
61                          try:
62                                  self.client.check_msg()
63                                  message = self.mMaker.
                                        temperatureMsg(values["
                                        Temperature"])
64                                  if message is not None:
65                                          self.publish(self.
                                                TOPIC_SUB4,message)
66                                  message = self.mMaker.humidityMsg(
                                        values["Humidity"])
67                                  if message is not None:
68                                          self.publish(self.
                                                TOPIC_SUB5,message)
69
70
71                          except OSError as e:
72                                  print("Errore OSError in check_msg:
                                        ", e)
73                                  try:
74                                          self.disconnect()
75                                  except:
76                                          pass
77                                  try:
78                                          self.client.set_callback(
                                                self.
                                                sub_callback_handler)
```

```
79                                          self.connect()
80                                          self.subscribes()
81
82                                          print("Riconnesso␣MQTT␣dopo
                                                ␣errore")
83                                          return 1
84
85                                  except Exception as e2:
86                                          print("Errore␣riconnessione
                                                ␣MQTT:", e2)
87                                          return 0
88                  return was_connected_MQTT
89
90
91          """ Restituisce l'insieme di topics """
92          def getSUB_TOPICS(self):
93                  return self.SUB_TOPICS
94
95
96          def connect(self):
97                  self.client.connect()
98
99
100         def disconnect(self):
101                 self.client.disconnect()
102
103
104         """ Pubblica su TOPIC il messaggio msg """
105         def publish(self, TOPIC, msg):
106                 self.client.publish(TOPIC, msg)
```

Listing 3: Questa classe gestisce la creazione del client MQTT. Il costruttore inizializza i vari topic e altre variabili di connessione quali indirizzo ip del broken e id del client. Infine inizializza l'oggetto client impostando la funzione di callback inviata come parametro di input. Il metodo checkAndReadMsg() pubblica i valori letti dal sensore di umidità e temperatura e risolve eventuali problemi di connessione.

3.4 File: KeyPad.py

```
1  class KeyPad:
2
3          """ Costruttore """
4          def __init__(self, in1, in2, in3, in4, out1, out2, out3 ,
               out4):
5
6                  """ Mappa dei tasti: righe per colonne """
7                  self.KEY_MAP = [
8                  ['1','2','3','A'],
9                  ['4','5','6','B'],
10                 ['7','8','9','C'],
11                 ['*','0','#','D']
```

```
12                  ]
13
14                  """ Definizione dei pin usati """
15                  self.rowPins = [Pin(p, Pin.IN, Pin.PULL_UP) for p
                        in [in1, in2, in3, in4]]
16                  self.colPins = [Pin(p, Pin.OUT) for p in [out1,
                        out2, out3, out4]]
17
18
19          """ Scan del tasto premuto """
20          def scan(self):
21                  for colNum, col in enumerate(self.colPins):
22                          """ Tutte le colonne a 1 tranne una (col)
                                """
23                          for c in self.colPins:
24                                  c.value(1)
25                          col.value(0)
26
27                          for rowNum, row in enumerate(self.rowPins):
28                                  if row.value() == 0:
29                                          time.sleep_ms(50)  # Si
                                                evita il debounce
30                                          if row.value() == 0:  #
                                                verifica di nuovo
31                                                  return self.KEY_MAP
                                                        [rowNum][colNum]
32                  return None
33
34          """ Lettura attiva del tasto premuto """
35          def lettura(self):
36                  key = self.scan()
37                  if key:
38                          #print('Hai premuto: ', key, ' di tipo: ',
                                type(key))
39                          while self.scan() == key:
40                                  time.sleep_ms(20)
41                          time.sleep_ms(50)
42                  return key
43
44          def letturaPin(self, oled, pos, num=4):
45                  password = ''
46
47                  for i in range(num):
48                          key = self.lettura()
49                          while key == None:
50                                  key = self.lettura()
51                          password = password + key
52                          #print('Hai premuto',key)
53                          pos = oled.write(pos[0],pos[1],0,' * ',
                                clean=False)
54                          oled.show()
```

```
55
56                  return password
```

Listing 4: Questa classe gestisce il tastierino utilizzato e fornisce alcuni metodi di lettura. Il costruttore inizializza la mappa dei pulsanti del tastierino sottoforma di matrice. Poi inizializza i pin di input (relativi alle 4 righe del tastierino) e i pin di output (relativi alle 4 colonne del tastierino). Il metodo scan() serve per identificare quale tasto è premuto. Scorre tutte le colonne impostandole a 0 una alla volta; va poi a scorrere ogni riga: se il valore è 0 significa che c'è un contatto in quella riga e quella colonna. È presente il controllo sul debounce di 50ms. Il metodo lettura() rende la lettura dei tasti affidabile evitando che, tenendo premuto un tasto, questo venga letto più volte. Il metodo letturaPin() serve a leggere un pin lungo 4 caratteri di default.