

UNIVERSITÀ DEGLI STUDI DI SALERNO

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Documentazione Progetto IoT

Annamaria Scermino, Anuar Zourhi, Gerardo Selce

TheBox

1 giugno 2025

Indice

1	Introduzione	2
2	Architettura del Sistema	2
3	Descrizione del Codice	2
3.1	File: <code>main.py</code>	2
3.2	File: <code>Handler.py</code>	4
3.3	File: <code>MQTTClass.py</code>	6

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

3.1 File: main.py

```
1 while True:
2
3     if stato == STATO_CONFIGURAZIONE_PIN:
4         pass
5         #
```

```
6         # AGGIORNAMENTO DELLO STATO
7         #
8         stato = NUOVO_STATO
9
10        elif stato == STATO_CONFIGURAZIONE_WIFI:
11            pass
12            #
13            # AGGIORNAMENTO DELLO STATO
14            #
15            stato = NUOVO_STATO
16
17        elif stato == STATO_CONNESSIONE:
18            pass
19            #
20            # AGGIORNAMENTO DELLO STATO
21            #
22            stato = NUOVO_STATO
23
24        elif stato == STATO_VISTA_MENU:
25            pass
26            #
27            # AGGIORNAMENTO DELLO STATO
28            #
29            stato = NUOVO_STATO
30
31        elif stato == STATO_CAMBIO_CONFIGURAZIONE:
32            pass
33            #
34            # AGGIORNAMENTO DELLO STATO
35            #
36            stato = NUOVO_STATO
37
38        elif stato == STATO_INSERTIMENTO_PIN:
39            pass
40            #
41            # AGGIORNAMENTO DELLO STATO
42            #
43            stato = NUOVO_STATO
44
45        elif stato == STATO_SBLOCCATO:
46            pass
47            #
48            # AGGIORNAMENTO DELLO STATO
49            #
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
5         =70.0, sogliaDis=10.0):
6         self.dht22 = dht22
7         self.hcsr04 = hcsr04
8         self.sogliaTemp = 0.0
9         self.sogliaHum = 0.0
10        self.sogliaDis = 0.0
11
12        if 'temp.txt' in os.listdir():
13            with open('temp.txt', 'r') as f:
14                self.sogliaTemp = float(f.read())
15        else:
16            with open('temp.txt', 'w') as f:
17                f.write(str(sogliaTemp))
18                self.sogliaTemp = sogliaTemp
19
20        if 'hum.txt' in os.listdir():
21            with open('hum.txt', 'r') as f:
22                self.sogliaHum = float(f.read())
23        else:
24            with open('hum.txt', 'w') as f:
25                f.write(str(sogliaHum))
26                self.sogliaHum = sogliaHum
27
28        if 'dis.txt' in os.listdir():
29            with open('dis.txt', 'r') as f:
30                self.sogliaDis = float(f.read())
31        else:
32            with open('dis.txt', 'w') as f:
33                f.write(str(sogliaDis))
34                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":
42                        dht_values[1]
43                    }
44     return dict_values
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

```

1 class MQTT:
2
3     """ Costruttore """
4     def __init__(self, sub_callback_handler):
5         """ Definizione variabili connessione MQTT """
6         self.CLIENT_ID    = 'esp32'
7         self.BROKER        = '192.168.131.51'
8         self.USER          = ''
9         self.PASSWORD      = ''
10        self.TOPIC         = b'TheBox'
11        self.TOPIC_SUB1    = b'TheBox/OpenCaveaux'
12        self.TOPIC_SUB2    = b'TheBox/CloseCaveaux'
13        self.TOPIC_SUB3    = b'TheBox/CaveauxStatus'
14        self.TOPIC_SUB4    = b''
15        self.TOPIC_SUB5    = b'TheBox/CaveauxStatus/Hum'
16        self.TOPIC_SUB6    = b'TheBox/Pin'
17        self.TOPIC_SUB7    = b'TheBox/Pin/WrongPassword'
18        self.TOPIC_SUB8    = b'TheBox/Allarme/Furto'
19        self.TOPIC_SUB9    = b'TheBox/Allarme/Risolto'
20        self.TOPIC_SUB10   = b'TheBox/Allarme/PinErrato'
21        self.TOPIC_SUB11   = b'TheBox/Status/Ping'
22        self.TOPIC_SUB12   = b'TheBox/Status/Pong'
23        self.TOPIC_SUB13   = b'TheBox/Allarme/StopBuzzer'
24        self.TOPIC_SUB14   = b'TheBox/CaveauxStatus/Temp/
25                               NuovaSoglia'
26        self.TOPIC_SUB15   = b'TheBox/CaveauxStatus/Hum/
27                               NuovaSoglia'
28        self.TOPIC_SUB16   = b'TheBox/CaveauxStatus/Dis/
29                               NuovaSoglia'
30        self.SUB_TOPICS    = [self.TOPIC_SUB1,
31                               self.TOPIC_SUB2, self.TOPIC_SUB3,
32                               self.TOPIC_SUB4, self.TOPIC_SUB5,
33                               self.TOPIC_SUB6, self.TOPIC_SUB7,
34                               self.TOPIC_SUB8, self.TOPIC_SUB9,
35                               self.TOPIC_SUB10, self.TOPIC_SUB11,
36                               self.TOPIC_SUB12, self.TOPIC_SUB13,
37                               self.TOPIC_SUB14, self.TOPIC_SUB15,
38                               self.TOPIC_SUB16]
39
40        self.SUB_TOPICS_SUB=[self.TOPIC_SUB2,
41                               self.TOPIC_SUB6, self.TOPIC_SUB11,
42                               self.TOPIC_SUB13, self.TOPIC_SUB14,
43                               self.TOPIC_SUB15, self.TOPIC_SUB16]
44
45        """ Inizializzazione dell'oggetto client """
46        self.client = MQTTClient(self.CLIENT_ID,
47                                   self.BROKER, user=self.USER,
48                                   password=self.PASSWORD, keepalive=60)
49        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                               ):
61             if wifi.isconnected() and self.client is not None
62                and was_connected_MQTT:
63                 try:
64                     self.client.check_msg()
65                     message = self.mMaker.
66                         temperatureMsg(values["
67                             Temperature"])
68                     if message is not None:
69                         self.publish(self.
70                             TOPIC_SUB4,message)
71                     message = self.mMaker.humidityMsg(
72                         values["Humidity"])
73                     if message is not None:
74                         self.publish(self.
75                             TOPIC_SUB5,message)
76
77                 except OSError as e:
78                     print("Errore_OSError_in_check_msg:
79                         ", e)
80                     try:
81                         self.disconnect()
82                     except:
83                         pass
84                     try:
85                         self.client.set_callback(
86                             self.
87                                 sub_callback_handler)
88                         self.connect()
89                         self.subscribes()
90
91                         print("Riconnesso_MQTT_dopo
92                             _errore")
93                         return 1
94
95                     except Exception as e2:
```



```
86         print("Errore_riconnessione
87             _MQTT:", e2)
88         return 0
89
90         return was_connected_MQTT
91
92     """ Restituisce l'insieme di topics """
93     def getSUB_TOPICS(self):
94         return self.SUB_TOPICS
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 broker 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.