University of Bremen

Final Project

Internet of Things

Theft Protection for IoT Devices

Submitted by:

Selim Hossain (3202203)

Zeeshan Rehman (3201414)

Md Mirsiadul Murshalin (6002553)

comnets

Contents

Summary:

The project is intended protect IoT device from being stolen. This is achieved by detecting such case in which device is being stolen and once the theft is detected/triggered, the user or control room will be alarmed and the device has actuators that will scare away the thief.

The IoT device may have been installed in a room, on the wall or outdoors in a remote location. In our proposed concept, the theft can be detected from the movement of the device. With that being said, the movement can be detested from the built-in Accelerometer sensor (in our case in Pysense module). This will send the trigger to the the Thing Network (TTN) server through Lora Gateway because the device is Lora-enabled. A control system (MQTT) integrates with the cloud server, which extracts the data from the things network server. This trigger will immediately be received at the control room (anywhere in the world) and the alarm will be activated and a SMS will also be sent to the designated phone number according to the logic.

Challenges:

1. **Prevent false alarm:** Initially, we received the raw data from the Accelerometer in the form of pitch and roll using the given Library LIS2HH12(). As the sensor is very sensitive, the data is constantly changing even if device resides still. To prevent that, we create a range of 10 % threshold. Suppose we the sensor gives pitch value 22 and 10% of 22 will be 2.2, So for sensor's new value it should be within range of 19.8 to 24.2. Secondly, we also have to prevent if someone mistakenly bumps against the device. In order to achieve that we measure two values each two seconds apart, we take the new data and compare it with the range of old value. If the new value is out of the range, it triggers that the device has been stolen. We initially struggle to determine what should be the threshold (in percentage). We therefore tried multiple times with various values through trial and error such a 1%, 2%, 5%, 20%, 50%, and eventually realized that the threshold value for our application is 10%.
2. **Adding device to TTN/Things network**: To connect with the TTN network, we first created an account on the website and attempted to access our data there. When we were unable, we realized we needed a gateway and contacted our professor for assistance. When our professor provided the gateway, we were able to join.
3. **Extracting Data from TTN Server:** To achieve this task, we used MQTT integration on TTN. When we received the data from the TTN server, we found that the data is in Json file and there are many payloads from different gateway. First, we find our gateway from the Json files and filter out the payload from this file.
4. **Connect the Display with The Raspberry pi:** We also faced problem to integrate the display with the Raspberry pi. When we connected the display with the Raspberry pi, we did not get any signal from the raspberry pi. We found that we need to install some library to the Raspberry pi to get the signal on the display therefore, we installed the specific library.

Hardware components:

- Pysense
- Lora GateWay
- Raspberry pi 3 (Installed at the control room)
- Alarm devices (Led, Buzzer, Display, Mobile Phone)
- Breadboard
- Jumper Wires

Software:

- VS Studio with Pymakr Extension
- Python programming

Pysense: A new sensor shield called Pysense is compatible with all Pycom multi-network modules. It is compatible with all Pycom boards and comes with a number of sensors, including ones for humidity, pressure, and ambient light.

Lora Gateway: Data transmitted through LoRaWAN radio is received by the LoRaWAN gateway, which digitizes it. The digital data from the sensor is enhanced with meta-data before being transmitted over a TCP/IP connection to the LoRaWAN network server. The sensors receive and transfer data from the LoRaWAN network server in a similar manner. Ethernet, WLAN, or cellular connections are all acceptable TCP/IP connections. Backhaul refers to this link to the LoRaWAN network server.

Raspberry pi: Raspberry Pi 3 Model B+ is 64-bit quad core processor running at 1.4GHz, dual-band 2.4GHzand 5GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT. The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market. The mechanical footprint of the Raspberry Pi 3 Model B+ is identical to that of the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model.

Alarm Devices:

1. **Led:** A light-emitting diode (LED) is a semiconductor light source that emits light when current flows through it.
2. **Buzzer**: The buzzer is a sounding device that can convert audio signals into sound signals. It is usually powered by DC voltage. It is widely used in alarms.
3. **Display:** 7 inch (approx. 17.7 cm) large LC display with HDMI connection.
4. **SMS:** An SMS will be sent to the designated phone number once the theft is triggered.

Architecture of the System:

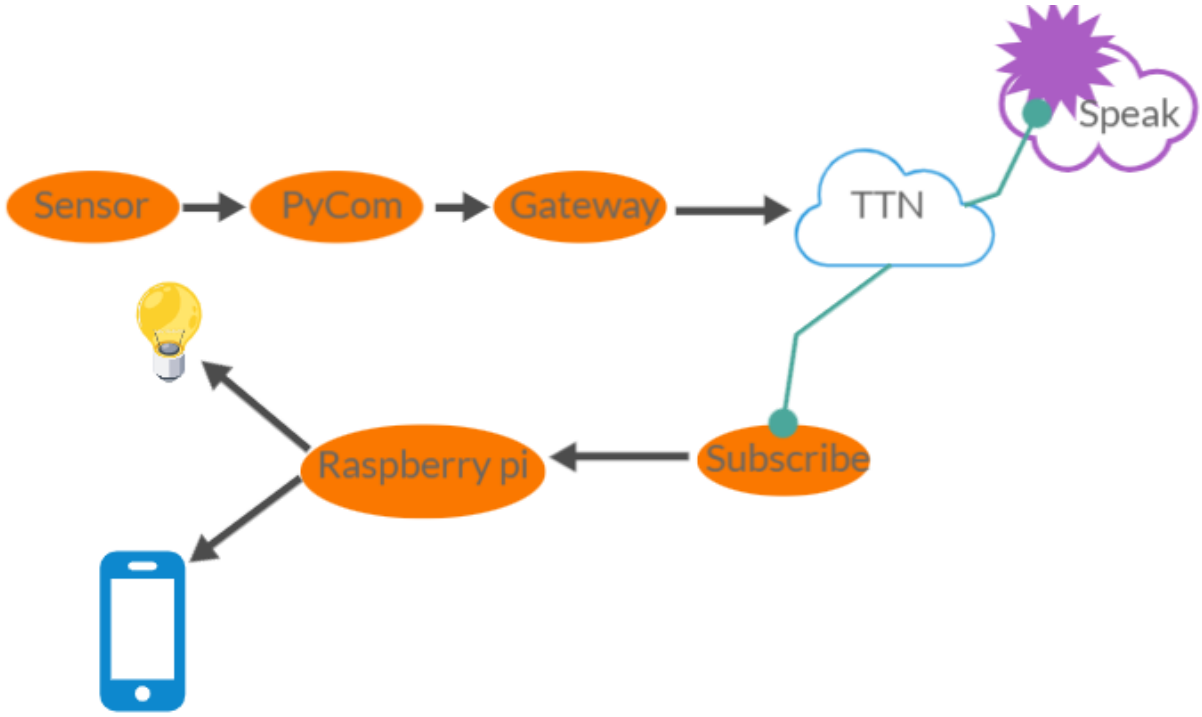

Figure 1. Architecture of the Solution

The fig.1 shows the architecture of our solution. In our project, we used Pysense module which have built-in accelerometer sensor which gives the roll and pitch value. we integrated Pysense lora-enabled device with the LoRa Gateway. When the theft is detected, we activated the stolen flag and send the data to the thing network through the gateway. We send the data to the Things Network server using Lora Network Protocol. We also integrated the thingspeak to the Thing Network for data visualization therefore, we change the uplink data format which is suitable for the thingspeak. To extract the data from the thing network we use MQTT (Message Queuing Telemetry Transport) protocol. After receiving the data, we filtered our payload and decode this payload for further use. We notify the owner by SMS and activate the buzzer and led when the flag of the output is high in our payload.

The Things Network (TTN):

Firstly, we create an account on The Things Network. In the next step we add an end device on TTN network and generate the Device and application parameters required for **OTAA** activation such as **AppEUI**, **DevEUI** and **AppKey** as shown in fig. 2.

Figure 2. End Device Parameters on TTN Server

The Thingspeak:

We used the thingspeak for the Visualization which is integrated on the Things Network website. We show the Roll, Pitch and status of the Output on the Thingspeak. As shown in fig. 3.

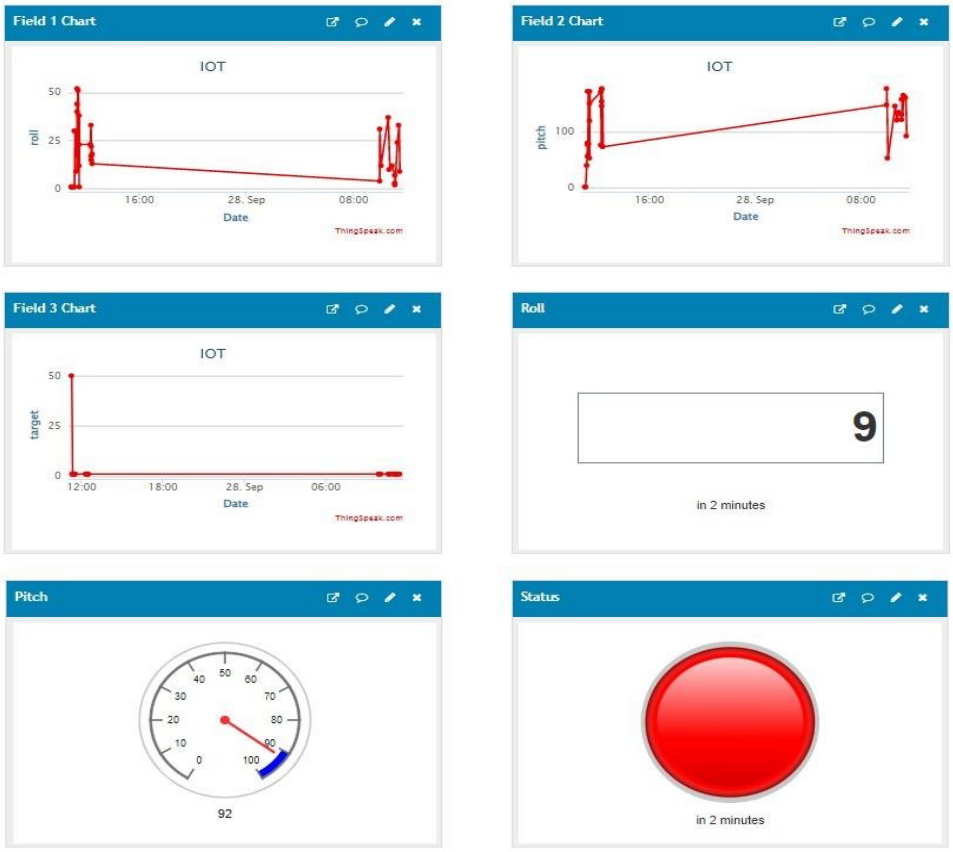

Figure 3. ThingsSpeak dashboard

Hardware Setup:

The fig. 4 shows our hardware setup.

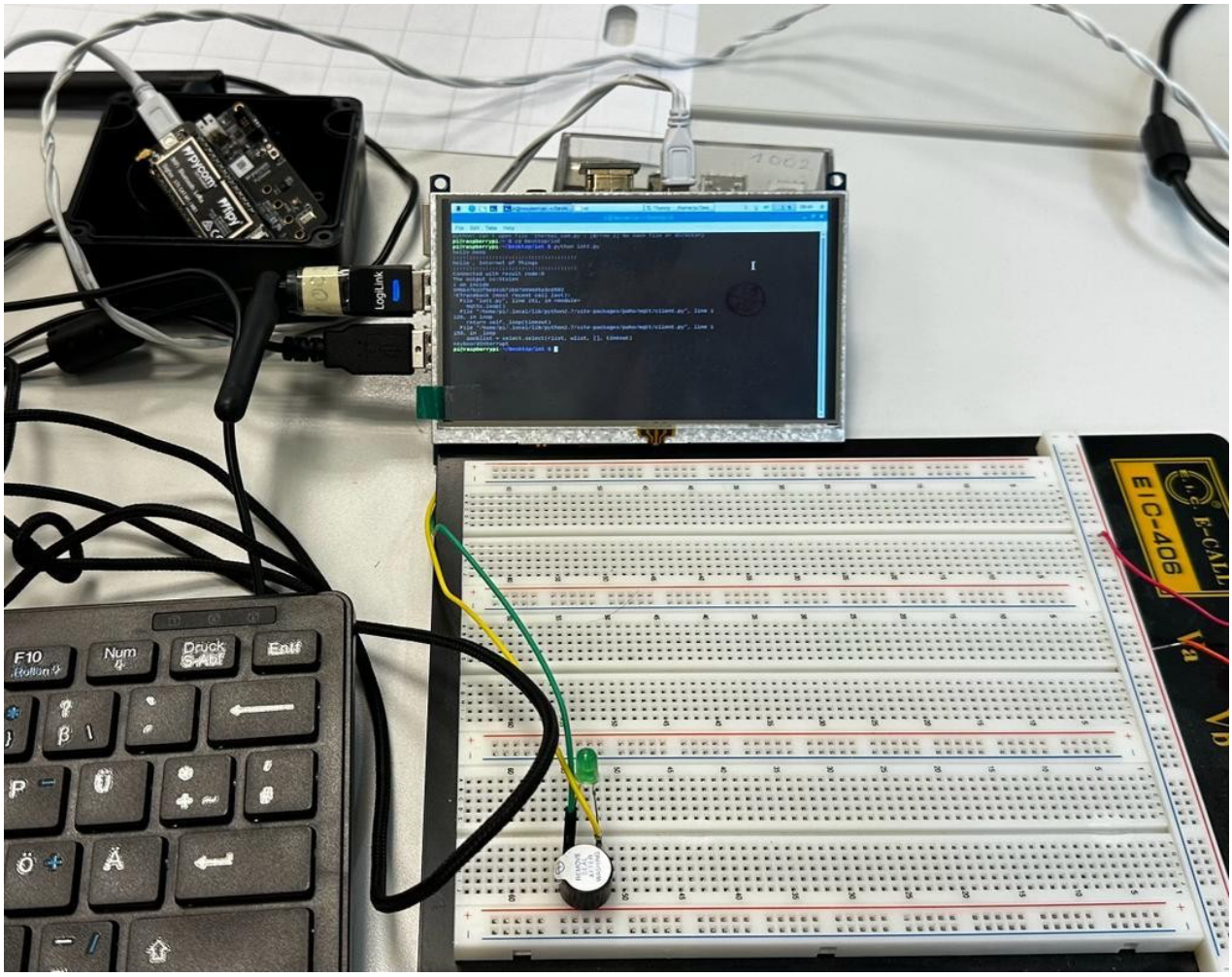

Figure 4. Hardware Setup

Summary:

Step-1: Connect the Pysense module to the laptop and upload the code through VS code.

Step-2: Send the trigger to TTN using Lora Gateway.

Step-3: Subscribe the sensor data form the Things Network using MQTT Protocol.

Step-4: Filter the payload from the Json file and extract the data according to the topics

Step-5: Run the program on the Raspberry pi and shows the output on the display, send a SMS to the operator and turn on the Buzzer and Led light to scare away the thief.

Future work:

If you consider what the future of anti-theft technology holds for users, IoT (Internet of Things) is considered important. The trend of IoT will make it tougher to steal high value devices, as they would come with embedded chips that can be used to monitor the location of the product with the internet. Following are few options possible.

1. Anti-theft software will make it possible for users to block a thief from stealing the device. Already, phones use apps that allow users to lock their device through the internet in case it gets stolen (like the Find My iPhone App on iOS devices). Clearly, such applications will make it possible to block the use of any product that's been stolen, not just phones or tablets.
2. If the device is in the vicinity of a WIFI router, the IoT device can monitor the connection signal strength with the network and if being carried away the signal strength will weaken and device can trigger theft status before losing the connection at all. This can be used as a backup option in case the Lora-Gateway is down.
3. An InfraRed (IR) sensor can be embedded into our application as another layer of protection.

Appendix:

Source Code:

```
from cmath import pi
from network import LoRa
from pycoproc_2 import Pycoproc
import socket
import time
import ustruct as struct
import ubinascii
from LIS2HH12 import LIS2HH12

acc = LIS2HH12()
py = Pycoproc()
lora = LoRa(mode=LoRa.LORAWAN, region=LoRa.EU868)

# create an OTAA authentication parameters, change them to the provided credentials
app_eui = ubinascii.unhexlify('0000000000000965')
app_key = ubinascii.unhexlify('20800EA7712A6FD1BD44471F95EEC6D9')
lora.join(activation=LoRa.OTAA, auth=(app_eui, app_key), timeout=0)
def send_data():

  # wait until the module has joined the network
  while not lora.has_joined():
    time.sleep(2.5)
    print('Not yet joined...')

  print('Joined')
  # create a LoRa socket
  s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)

  # set the LoRaWAN data rate
  s.setsockopt(socket.SOL_LORA, socket.SO_DR, 5)

  # make the socket blocking
  # (waits for the data to be sent and for the 2 receive windows to expire)
  s.setblocking(True)

  #va = struct.pack(str)
  s.send(bytes([0x01]))
```

```
while True:

  pitch = abs(int(acc.pitch()))
  roll = abs(int(acc.roll()))
  time.sleep(3)
  new_pitch = abs(int(acc.pitch()))
  new_roll = abs(int(acc.roll()))
  pitch_over_head = int(pitch * 0.1)
  roll_over_head = int(roll * 0.1)
  #print(perc_10)

  if new_pitch not in range(pitch - pitch_over_head, pitch + pitch_over_head+1):
    if new_roll not in range(roll - roll_over_head, roll + roll_over_head+1):
      print("stolen")
      send_data()
  else:

    print("okay")
```