**LAB: การอ่านข้อมูลเซนเซอร์จากนาฬิกาอัจฉริยะผ่าน BLE**

# **จุดประสงค์**

1. เพื่อศึกษาการอ่านข้อมูลเซนเซอร์จากนาฬิกาอัจฉริยะ
2. เพื่อศึกษาและทำความเข้าใจหลักการทำงานของเซนเซอร์ที่อยู่ในนาฬิกาอัจฉริยะ
3. เพื่อศึกษาตัวอย่างการนำข้อมูลเซนเซอร์ที่ได้จากนาฬิกามาทำการวิเคราะห์ข้อมูล

# **การอ่านข้อมูลเซนเซอร์จากนาฬิกาอัจฉริยะ**

สำหรับการทดลองนี้เป็นการทดลองดึงข้อมูลดิบในส่วนของข้อมูลจาก เซนเซอร์วัดความเร่ง (Accelerometer) แบบ 3 แกน เซนเซอร์วัดความเร็วเชิงมุม (Gyroscope) เซนเซอร์ความดันบรรยากาศ (Air pressure) และเซนเซอร์วัดอัตราการเต้นหัวใจ (ฺHeart rate) และเซนเซอร์วัดระดับออกซิเจนในเลือด (Spo2) หลักการทำงานของโปรแกรม เราจะใช้คอมพิวเตอร์เชื่อมต่อนาฬิกาผ่าน บลูทูธ จากนั้นเมื่อตัวนาฬิกาได้รับการเชื่อมต่อแล้วตัวนาฬิกาจะทำการ Steam ข้อมูลออกทาง UUID หลายเลข

| 6E400003-B5A3-F393-E0A9-E50E24DCCA9E |
| --- |



โดยแบ่งกลุ่มของเซนเซอร์ออกเป็น 2 กลุ่มตามประเภทของข้อมูลดังนี้

1. ข้อมูลการเคลื่อนไหว ( Movement) ซึ่งจะใช้ข้อมูลจาก เซนเซอร์วัดความเร่ง (Accelerometer) แบบ 3 แกน เซนเซอร์วัดความเร็วเชิงมุม (Gyroscope) และข้อมูลจากเซนเซอร์ความดันบรรยากาศ (Air pressure)
2. ข้อมูลสัญญาณชีพ (Vital signs) ซึ่งจะเป็นข้อมูลจาก เซนเซอร์วัดอัตราการเต้นหัวใจ (ฺBPM)และเซนเซอร์วัดระดับออกซิเจนในเลือด (Spo2)

โดยแต่ละประเภทมีจะถูกส่งออกมาโดยมีรูปแบบ Protocol ดังต่อไปนี้

Package Structure

| typedef struct \_\_attribute\_\_((packed)){  uint8\_t header;  uint16\_t seq\_number;  int16\_t acc\_x;  int16\_t acc\_y;  int16\_t acc\_z;  int16\_t gyro\_x;  int16\_t gyro\_y;  int16\_t gyro\_z;  float air\_pressure;  }acc\_data\_steam\_t;  typedef struct \_\_attribute\_\_((packed)){  uint8\_t header;  uint8\_t hr;  uint8\_t hr\_conf;  uint8\_t spo2;  uint8\_t spo2\_conf;  uint8\_t scd\_contact\_state;  uint32\_t raw\_data\_led1;  uint16\_t skin\_temp;  }hr\_spo2\_data\_steam\_t; |
| --- |

**คำอธิบายรูปแบบ Package ของข้อมูล acc\_data\_steam\_t**

สำหรับ Package นี้จะทำหน้าที่เก็บข้อมูลการเคลื่อนไหวของผู้สวมใส่โดยมีรายละเอียดดังต่อไปนี้

| **ชื่อตัวแปล** | **ขนาด (byte)** | **คำอธิบาย** |
| --- | --- | --- |
| header | 1 | เป็นตัวกำหนดว่า package นั้นเป็นข้อมูลชนิดการเคลื่อนไหวโดยจะมีค่าเป็น 0xFF |
| seq\_number | 2 | เป็นตัวแปรที่จะบอกค่า sequence number ของข้อมูลเพื่อทำให้เราทราบว่ามีข้อมูลขาดหายไปหรือไม่ |
| acc\_x | 2 | ข้อมูลจากเซนเซอร์วัดความเร่งในแนวแกน x โดยจะมีค่าตั้งแต่ -32768 ถึง 32767 |
| acc\_y | 2 | ข้อมูลจากเซนเซอร์วัดความเร่งในแนวแกน y โดยจะมีค่าตั้งแต่ -32768 ถึง 32767 |
| acc\_z | 2 | ข้อมูลจากเซนเซอร์วัดความเร่งในแนวแกน z โดยจะมีค่าตั้งแต่ -32768 ถึง 32767 |
| gyro\_x | 2 | ข้อมูลจากเซนเซอร์วัดความเร็วเชิงมุมในแนวแกน x โดยจะมีค่าตั้งแต่ -32768 ถึง 32767 |
| gyro\_y | 2 | ข้อมูลจากเซนเซอร์วัดความเร็วเชิงมุมในแนวแกน y โดยจะมีค่าตั้งแต่ -32768 ถึง 32767 |
| gyro\_z | 2 | ข้อมูลจากเซนเซอร์วัดความเร็วเชิงมุมในแนวแกน z โดยจะมีค่าตั้งแต่ -32768 ถึง 32767 |
| air\_pressure | 4 | ข้อมูลจากเซนเซอร์วัดความดันบรรยากาศ |

**คำอธิบายรูปแบบ Package ของข้อมูล hr\_spo2\_data\_steam\_t**

สำหรับ Package นี้จะทำหน้าที่เก็บข้อมูลสัญญาณชีพของผู้สวมใส่โดยมีรายละเอียดดังต่อไปนี้

| **ชื่อตัวแปล** | **ขนาด (byte)** | **คำอธิบาย** |
| --- | --- | --- |
| header | 1 | เป็นตัวกำหนดว่า package นั้นเป็นข้อมูลสัญญาณชีพโดยจะมีค่าเป็น 0xFE |
| hr | 1 | ข้อมูลจากเซนเซอร์คำนวณค่าอัตราการเต้นหัวใจ โดยจะมีค่าตั้งแต่ 40 ถึง 255 |
| hr\_conf | 1 | ข้อมูลที่จะค่อยบอกว่าค่าอัตราการเต้นหัวใจ มีความมั่นใจว่าจะถูกต้องกี่เปอร์เซ็น โดยจะมีค่าตั้งแต่ 0% ถึง 100% |
| spo2 | 1 | ข้อมูลจากเซนเซอร์คำนวณค่าระดับออกซิเจนในเลือด โดยจะมีค่าตั้งแต่ 0% ถึง 100% |
| spo2\_conf | 1 | ข้อมูลที่จะค่อยบอกว่าค่าระดับออกซิเจนในเลือดที่ได้นั้น มีความมั่นใจว่าจะถูกต้องกี่เปอร์เซ็น โดยจะมีค่าตั้งแต่ 0% ถึง 100% |
| scd\_contact\_state | 1 | ข้อที่จะบ่งบอกว่า ณ ขณะนั้นมีการสวมใส่นาฬิกาอยู่หรือไม่โดยที่  0x01 คือไม่มีการสวมใส่อุปกรณ์  0x03 คือมีการสวมใส่อุปกรณ์อยู่ ณ ขณะนั้น |
| raw\_data\_led1 | 4 | ข้อมูลดิบจากเซนเซอร์วัดความเข้มแสง |
| skin\_temp | 2 | ข้อมุลจากเซนเซอร์ตรวจวัดอุณหภูมิ |

## อุปกรณ์การทดลอง

1. คอมพิวเตอร์ที่มีระบบปฎิบัติการ Windows หรือ Linux หรือ OS X
2. นาฬิกาอัจฉริยะรุ่น Hive Smart พร้อมสายชาร์จแบตเตอรี่

## ขั้นตอนการทดลอง

### ติดตั้ง python และ package ที่เกี่ยวข้อง

#### 1. ทำการติดตั้ง Python3

สามารถโหลดได้จากลิงค์ด้านล่างนี้

| <https://www.python.org/downloads/> |
| --- |

#### 2. ทำการติดตั้ง PIP

สำหรับระบบปฏิบัติการ windows

ทำการ Download ไฟล์จากลิงค์ต่อไปนี้ [Link](https://bootstrap.pypa.io/get-pip.py) จากนั้นไปที่ Folder ที่ Download มาและทำการเปิด CMD ต่อจากนั้นทำการใช้คำสั่งต่อไปนี้ในการติดตั้ง

| $ python get-pip.py |
| --- |

สำหรับระบบปฏิบัติการ Linux

| $ sudo apt-get install python3-pip |
| --- |

สำหรับระบบปฏิบัติ macOS เนื่องจากมีการติดตั้ง python ไว้อยู่แล้วดังสามารถใช้คำสั่งดังต่อไปนี้ได

| $ sudo python get-pip.py |
| --- |

หรือจะใช้คำสั่งต่อไปนี้แทนก็สามารถทำได้

| $ sudo easy\_install pip |
| --- |

#### 3. ทำการติดตั้ง Package Python สำหรับใช้พัฒนาดังต่อไปนี้

คำสั่งติดตั้ง Package [numpy](https://pypi.org/project/numpy/)

เป็น package ที่ใช้สำหรับการเรียกใช้การคำนวณที่เป็น อาเรย์ ต่าง ๆ ของ python

| $ pip install numpy |
| --- |

คำสั่งติดตั้ง Package [Bleak](https://pypi.org/project/bleak/)  ใช้สำหรับการเขียน หรืออ่านข้อมูลและการแจ้งเตือน ด้วยบูลทูธ

$ pip install bleak

## LAB1: ข้อมูลการเคลื่อนไหว ( Movement)

ทำการสร้างไฟล์ที่ชื่อว่า lab1\_1\_hive\_smart.py และทำการพิมพ์ source code ต่อไปนี้ลงในไฟล์ดังกล่าว

| # -\*- coding: utf-8 -\*-  """  Notifications  -------------  Example showing how to add notifications to a characteristic and handle the responses.  Updated on 2019-07-03 by hbldh <henrik.blidh@gmail.com>  """  #from ctypes import HRESULT  import sys  import asyncio  import platform  import numpy as np  from struct import \*  from bleak import BleakClient  import time  from datetime import datetime    datetime\_format = "%m/%d/%Y %H:%M:%S"  HEADER ="\t seq\_number, \tacc\_x, acc\_y,acc\_z, \t\tgyro\_x, gyro\_y,gyro\_z ,Timestamp"  # you can change these to match your device or override them from the command line  CHARACTERISTIC\_UUID = "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"  ADDRESS = (  "F8:6F:87:C1:82:04"  if platform.system() != "Darwin"  else "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"  )  def notification\_handler(sender, data):  """Simple notification handler which prints the data received."""  #print("{0}: {1}".format(sender, data))  #print(data[0])    if data[0] == 0xFF:  seq\_number = (data[2]<<8) | data[1]  acc\_x = np.int16( (data[4]<<8) | data[3] )  acc\_y = np.int16( (data[6]<<8) | data[5] )  acc\_z = np.int16( (data[8]<<8) | data[7] )  gyro\_x = np.int16( (data[10]<<8) | data[9] )  gyro\_y = np.int16( (data[12]<<8) | data[11] )  gyro\_z = np.int16( (data[14]<<8) | data[13] )  air\_pressure = unpack('<f', data[15:19])  time\_now = datetime.now()  timestamp = time\_now.strftime(datetime\_format)  print(  "\t "+str(seq\_number)  +',\t\t'+str(acc\_x)  +',\t'+str(acc\_y)  +',\t'+str(acc\_z)  +',\t\t'+str(gyro\_x)  +',\t'+str(gyro\_y)  +',\t'+str(gyro\_z)  +',\t'+timestamp  )    async def main(address, char\_uuid):  async with BleakClient(address) as client:  print(f"Connected: {client.is\_connected}")  print('Movement')  print (HEADER)    await client.start\_notify(char\_uuid, notification\_handler)  await asyncio.sleep(5.0)  await client.stop\_notify(char\_uuid)  print("stop")      if \_\_name\_\_ == "\_\_main\_\_":  asyncio.run(  main(  sys.argv[1] if len(sys.argv) > 1 else ADDRESS,  sys.argv[2] if len(sys.argv) > 2 else CHARACTERISTIC\_UUID,  )  ) |
| --- |

ทำการรันโปรแกรมโดยใช้คำสั่งต่อไปนี้และทำการบันทึกผล

| $ python lab1\_1\_hive\_smart.py |
| --- |

## บันทึกผลการทดลอง

1. output ข้อมูลที่ได้กรณีวางนาฬิกาไว้เฉยๆ กับโต๊ะ
2. output ข้อมูลที่ได้กรณีสวมใส่นาฬิกาจากนั้นเดินไปมา

## LAB2: การอ่านข้อมูล สัญญาณชีพ (Vital signs)

ทำการสร้างไฟล์ที่ชื่อว่า lab1\_2\_hive\_smart.py และทำการพิมพ์ source code ต่อไปนี้ลงในไฟล์ดังกล่าว

| # -\*- coding: utf-8 -\*-  import sys  import asyncio  import platform  from struct import \*  from bleak import BleakClient  import time  from datetime import datetime  datetime\_format = "%m/%d/%Y %H:%M:%S"  HEADER = "\tHR, Hr\_conf, Spo2, Spo2\_conf, Scd\_contact\_state, Raw\_data\_led1, Skin\_temp, Timestamp"  # you can change these to match your device or override them from the command line  CHARACTERISTIC\_UUID = "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"  ADDRESS = (  "F8:6F:87:C1:82:04"  if platform.system() != "Darwin"  else "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"  )  def notification\_handler(sender, data):  """Simple notification handler which prints the data received."""  # print("{0}: {1}".format(sender, data))  # print(data[0])    if data[0] == 0xFE:  hr = data[1]  hr\_conf = data[2]  spo2 = data[3]  spo2\_conf = data[4]  scd\_contact\_state = data[5]  raw\_data\_led1 = (data[9]<<24) | (data[8]<<16) | (data[7]<<8) | data[6]  skin\_temp = ((data[11]<<8) | data[10] ) / 10.0    time\_now = datetime.now()  timestamp = time\_now.strftime(datetime\_format)    print(  '\t'+str(hr)  +'\t'+str(hr\_conf)  +',\t'+str(spo2)  +',\t'+str(spo2\_conf)  +',\t\t'+str(scd\_contact\_state)  +',\t\t'+str(raw\_data\_led1)  +',\t'+str(skin\_temp)  +',\t '+timestamp  )    async def main(address, char\_uuid):  async with BleakClient(address) as client:  print(f"Connected: {client.is\_connected}")  print('vital\_signs')  print (HEADER)    await client.start\_notify(char\_uuid, notification\_handler)  await asyncio.sleep(5.0)  await client.stop\_notify(char\_uuid)  print("stop")      if \_\_name\_\_ == "\_\_main\_\_":  asyncio.run(  main(  sys.argv[1] if len(sys.argv) > 1 else ADDRESS,  sys.argv[2] if len(sys.argv) > 2 else CHARACTERISTIC\_UUID,  )  ) |
| --- |

ทำการรันโปรแกรมโดยใช้คำสั่งต่อไปนี้และทำการบันทึกผล

| $ python lab1\_2\_hive\_smart.py |
| --- |

## บันทึกผลการทดลอง

1. output ข้อมูลที่ได้กรณีวางนาฬิกาไว้เฉยๆ กับโต๊ะ
2. output ข้อมูลที่ได้กรณีสวมใส่นาฬิกาจากนั้นให้เปรียบเทียบค่าต่าง ๆ ที่แสดงบน pc กับ นาฬิกา