*장현주\_2023011923*

|  |
| --- |
| **2024 ALTIS SW 개인 활동 보고서** |

|  |  |
| --- | --- |
| **활동 개요** | (통신) 3D시각화, GUI역확분담 |
| **일시** | 2024년 5월 19일 00시 |
| **작성자** | 김민주, 장현주 |
| **활동 내용** | *※ 어떤 활동을 진행하였는지 자세히 쓰기*  **3D시각화**    **GUI 역확분담**  전에 작성하였던 그림을 바탕으로 지도 맵은 민주언니가, 카메라는 현주가 구현해 보기로 역할을 나누었습니다.  또한 그래프나 배치, 3D같은 경우에는 같이 구현할 예정입니다. 추가적으로 구현이 필요한 경우에도 같이 구현할 예정입니다.  (지도와 그래프를 구현)  PyQt: Qt의 레이아웃에 Python의 코드를 연결하여 GUI 프로그램을 만들 수 있게 해주는 프레임워크으로 이를 사용하여 구현할 것입니다.  (명령 프롬프트에 ‘pip install pyserial pyqtgraph PyQt5 folium opencv-python’ (그래프)/’ pip install pyserial matplotlib PyQt5 folium ‘(지도), ‘pip install pyserial matplotlib PyQt5 folium opencv-python’(카메라)를 입력하여 설치합니다.)  MainWIndow클래스로 PyQt5를 사용하여 GUI를 구성합니다.  (지도)  앞서 PyQt에서 언급한 ‘folium’라이브러리를 설치하여 지도 위에 데이터를 표시할 수 있게 합니다.  mpa\_view변수에 Folium으로 생성한 지도 객체를 저장합니다.  지도를 생성하면 서울 위치를 중심으로 생성되기 때문에 > Map 객체를 생성할 때 location 매개변수를 사용하여 해당 위치의 위도(latitude)와 경도(longitude)를 지정합니다.  (고도그래프)  고도계 센서인 BMP280가 필요합니다.  ‘pygraph’ 라이브러리를 설치하여 실시간 고도 데이터를 그래프로 표시할 예정입니다.  QLabel 위젯을 추가하여 고도, 기압, 온도 데이터를 텍스트로 표시할 예정입니다.  (카메라)  OpenCV를 사용하여 웹캠 스트림을 캡처하고, PyQt5와 연동하여 이미지를 업데이트합니다. |
| **활동 사진** | |  | | --- | | altitude | | import pyqtgraph as pg  import numpy as np  class graph\_altitude(pg.PlotWidget):      def \_\_init\_\_(self):          super().\_\_init\_\_()          self.setTitle('Altitude', color='w', size='18pt')          self.setLabel('left', 'altitude (m)')          self.setLabel('bottom', 'Time(s)')          #graph style          pen = pg.mkPen(color=(255, 0, 127), width=2)          #set graph data          self.x = np.linspace(0,1,20) #0에서 1까지의 값을 가져오는데, 동일한 간격으로 20개 가져옴.          self.y = np.linspace(0,0,20) #0을 20개 갖고 있음          self.plot = self.plot(self.x, self.y, pen=pen)        def update(self, value): #value: 0에서 10까지 수 중 랜덤으로 하나! if 첫값 3          #x = 0, 0.05, 0.1 ...... 1 총 20개          #y = 0,0,0,0,0,.... 20개          x = self.x[-1]          self.x[:-1] = self.x[1:]          self.x[-1] = x+0.3 #업데이트 함수가 매 0.3초마다 업데이트하기 때문 0.3sec          self.y[:-1] = self.y[1:] #데이터 테이블 이동          self.y[-1] = float(value) #마지막에 새로운 데이터 추가          self.plot.setData(self.x, self.y) | | Groupbutton | | ## Ex 5-10. QGroupBox.  import sys  from PyQt5.QtWidgets import (QApplication, QWidget, QGroupBox, QRadioButton  , QCheckBox, QPushButton, QMenu, QGridLayout, QVBoxLayout)  class group\_button(QWidget):      def \_\_init\_\_(self):          super().\_\_init\_\_()          self.initUI()      def initUI(self):          grid = QGridLayout()          grid.addWidget(self.createFirstExclusiveGroup(), 0, 0)          grid.addWidget(self.createSecondExclusiveGroup(), 1, 0)          grid.addWidget(self.createNonExclusiveGroup(), 0, 1)          grid.addWidget(self.createPushButtonGroup(), 1, 1)          self.setLayout(grid)          self.setWindowTitle('Box Layout')          self.setGeometry(300, 300, 480, 320)          self.show()      def createFirstExclusiveGroup(self):          groupbox = QGroupBox('Exclusive Radio Buttons')          radio1 = QRadioButton('Radio1')          radio2 = QRadioButton('Radio2')          radio3 = QRadioButton('Radio3')          radio1.setChecked(True)          vbox = QVBoxLayout()          vbox.addWidget(radio1)          vbox.addWidget(radio2)          vbox.addWidget(radio3)          groupbox.setLayout(vbox)          return groupbox      def createSecondExclusiveGroup(self):          groupbox = QGroupBox('Exclusive Radio Buttons')          groupbox.setCheckable(True)          groupbox.setChecked(False)          radio1 = QRadioButton('Radio1')          radio2 = QRadioButton('Radio2')          radio3 = QRadioButton('Radio3')          radio1.setChecked(True)          checkbox = QCheckBox('Independent Checkbox')          checkbox.setChecked(True)          vbox = QVBoxLayout()          vbox.addWidget(radio1)          vbox.addWidget(radio2)          vbox.addWidget(radio3)          vbox.addWidget(checkbox)          vbox.addStretch(1)          groupbox.setLayout(vbox)          return groupbox      def createNonExclusiveGroup(self):          groupbox = QGroupBox('Non-Exclusive Checkboxes')          groupbox.setFlat(True)          checkbox1 = QCheckBox('Checkbox1')          checkbox2 = QCheckBox('Checkbox2')          checkbox2.setChecked(True)          tristatebox = QCheckBox('Tri-state Button')          tristatebox.setTristate(True)          vbox = QVBoxLayout()          vbox.addWidget(checkbox1)          vbox.addWidget(checkbox2)          vbox.addWidget(tristatebox)          vbox.addStretch(1)          groupbox.setLayout(vbox)          return groupbox      def createPushButtonGroup(self):          groupbox = QGroupBox('Push Buttons')          groupbox.setCheckable(True)          groupbox.setChecked(True)          pushbutton = QPushButton('Normal Button')          togglebutton = QPushButton('Toggle Button')          togglebutton.setCheckable(True)          togglebutton.setChecked(True)          flatbutton = QPushButton('Flat Button')          flatbutton.setFlat(True)          popupbutton = QPushButton('Popup Button')          menu = QMenu(self)          menu.addAction('First Item')          menu.addAction('Second Item')          menu.addAction('Third Item')          menu.addAction('Fourth Item')          popupbutton.setMenu(menu)          vbox = QVBoxLayout()          vbox.addWidget(pushbutton)          vbox.addWidget(togglebutton)          vbox.addWidget(flatbutton)          vbox.addWidget(popupbutton)          vbox.addStretch(1)          groupbox.setLayout(vbox)          return groupbox | | Visual3D | | # class visual3D:  #     def \_\_init\_\_(self):  #         pg.init()  #         display = (800, 600)  #         pg.display.set\_mode(display, DOUBLEBUF | OPENGL)  #         self.clock = pg.time.Clock()  #         gluPerspective(45, (display[0] / display[1]), 0.1, 50.0)  #         glTranslatef(0.0, 0.0, -5)  #         self.com = Communication('COM6', 115200)  #         self.xangle1 = 0  #         self.yangle1 = 0  #         self.zangle1 = 0  #         self.mainLoop()  #     def mainLoop(self):  #         running = True  #         while (running):  #             for event in pg.event.get():  #                 if (event.type == pg.QUIT):  #                     running = False  #             glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)  #             angles = self.UpdateAngle()  #             if angles:  #                 self.xangle1, self.yangle1, self.zangle1 = angles  #                 # Reset the matrix and apply new rotations  #                 glLoadIdentity()  #                 gluPerspective(45, (800 / 600), 0.1, 50.0)  #                 glTranslatef(0.0, 0.0, -5)  #                 glRotatef(self.xangle1, 1, 0, 0)  #                 glRotatef(self.yangle1, 0, 1, 0)  #                 glRotatef(self.zangle1, 0, 0, 1)  #                 DrawRocket()    #             glLoadIdentity()  #             gluPerspective(45, (800 / 600), 0.1, 50.0)  #             glTranslatef(0.0, 0.0, -5)  #             glRotatef(self.xangle1, 1, 0, 0)  #             glRotatef(self.yangle1, 0, 1, 0)  #             glRotatef(self.zangle1, 0, 0, 1)  #             DrawRocket()  #             pg.display.flip()  #             self.clock.tick(30)  #         self.quit()  #     def quit(self):  #         pg.quit()  #     def UpdateAngle(self):  #         if (self.com.isOpen()):  #             try:  #                 data\_table = self.com.getData()  #                 if (len(data\_table) == 4):  #                     print(data\_table)  #                     xangle = float(data\_table[1])  #                     yangle = float(data\_table[2])  #                     zangle = float(data\_table[3])  #                     return xangle, yangle, zangle  #             except ValueError:  #                 pass  # value error for Serial communication error    import pygame as pg  from pygame.locals import \*  from OpenGL.GL import \*  from OpenGL.GLU import \*  import serial  from datetime import datetime  import sys  from PyQt5.QtWidgets import \*  from PyQt5.QtCore import QTimer  from PyQt5.QtOpenGL import \*  def DrawRocket():      body\_length = 2.0      body\_radius = 0.2      cone\_height = 0.6      cone\_radius = 0.2      # 로켓 몸체 (원기둥)      glPushMatrix()      glTranslatef(0, 0, -body\_length / 2)      glColor3f(0.8, 0.8, 0.8)  # Light gray      quadric = gluNewQuadric()      gluCylinder(quadric, body\_radius, body\_radius, body\_length, 32, 32)      glPopMatrix()      # 로켓 머리 (원뿔)      glPushMatrix()      glTranslatef(0, 0, body\_length / 2)      glColor3f(1.0, 0.0, 0.0)  # Red      quadric = gluNewQuadric()      gluCylinder(quadric, cone\_radius, 0.0, cone\_height, 32, 32)      glPopMatrix()      # 로켓 꼬리 (디스크)      glPushMatrix()      glTranslatef(0, 0, -body\_length / 2)      glRotatef(180, 1, 0, 0)      glColor3f(0.0, 0.0, 1.0)  # Blue      gluDisk(quadric, 0.0, body\_radius, 32, 32)      glPopMatrix()  class Communication():        def \_\_init\_\_(self, port : str, baudrate:int):          self.port = port          self.baudrate = baudrate          self.ser = serial.Serial(port,baudrate)        def getData(self):          if (self.ser.inWaiting() > 0):                value = self.ser.readline()              #value = self.sio.readline()              decoded = str(value[0:len(value) - 2].decode("utf-8"))              value\_table = decoded.split(",")              value\_table.insert(0, self.getDate())          else:              value\_table = []            return value\_table      def getDate(self):          now = datetime.now()          date = now.strftime("%Y-%m-%d %H:%M:%S")          return date        def isOpen(self):          return self.ser.isOpen()    class visual3D(QGLWidget):      def \_\_init\_\_(self, parent=None):          super(visual3D, self).\_\_init\_\_(parent)          self.com = Communication('COM6', 115200)          self.angles = [0, 0, 0]          self.timer = QTimer(self)          self.timer.timeout.connect(self.updateGL)          self.timer.start(30)      def initializeGL(self):          glEnable(GL\_DEPTH\_TEST)          glMatrixMode(GL\_PROJECTION)          gluPerspective(45, (800 / 600), 0.1, 50.0)          glMatrixMode(GL\_MODELVIEW)          glTranslatef(0.0, 0.0, -5)      def paintGL(self):          glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)          glLoadIdentity()          gluPerspective(45, (800 / 600), 0.1, 50.0)          glTranslatef(0.0, 0.0, -5)          angles = self.UpdateAngle()          if angles:              self.angles = angles          glRotatef(self.angles[0], 1, 0, 0)          glRotatef(self.angles[1], 0, 1, 0)          glRotatef(self.angles[2], 0, 0, 1)          DrawRocket()      def UpdateAngle(self):          if self.com.isOpen():              try:                  data\_table = self.com.getData()                  if len(data\_table) == 4:                      xangle = float(data\_table[1])                      yangle = float(data\_table[2])                      zangle = float(data\_table[3])                      return xangle, yangle, zangle              except ValueError:                  pass          return self.angles | | a2 , a3 ( 이름이랑 변수만 다름 ) | | import pyqtgraph as pg  import numpy as np  class graph\_a2(pg.PlotWidget):        def \_\_init\_\_(self):          super().\_\_init\_\_()          self.setTitle('Dummy data', color='w', size='18pt')          self.setLabel('left', 'dummy data (unit)')          self.setLabel('bottom', 'Time(s)')          #graph style          pen = pg.mkPen(color=(255, 0, 127), width=2)          #set graph data          self.x = np.linspace(0,1,20) #0에서 1까지의 값을 가져오는데, 동일한 간격으로 20개 가져옴.          self.y = np.linspace(0,0,20) #0을 20개 갖고 있음          self.plot = self.plot(self.x, self.y, pen=pen)        def update(self, value): #value: 0에서 10까지 수 중 랜덤으로 하나! if 첫값 3          #x = 0, 0.05, 0.1 ...... 1 총 20개          #y = 0,0,0,0,0,.... 20개          x = self.x[-1]          self.x[:-1] = self.x[1:]          self.x[-1] = x+0.3 #업데이트 함수가 매 0.3초마다 업데이트하기 때문 0.3sec          self.y[:-1] = self.y[1:] #데이터 테이블 이동          self.y[-1] = float(value) #마지막에 새로운 데이터 추가          self.plot.setData(self.x, self.y) | | guiMain | | import sys  from PyQt5.QtWidgets import \*  from PyQt5.QtGui import QIcon  from PyQt5.QtCore import QTimer, QDateTime  import random  from visual3D import \*  from altitude import \*  from a3 import \*  from a2 import \*  from groupbutton import \*  class guiMain(QMainWindow):      def \_\_init\_\_(self):          super().\_\_init\_\_()          self.initUI()      def initUI(self):          self.setWindowTitle('GUI')          self.setWindowIcon(QIcon('.vscode/windowicon.jpg'))          self.setGeometry(150, 100, 2000, 1500)          # Timer to update the status bar with the current date and time          self.timer = QTimer(self)          self.timer.timeout.connect(self.updateStatusBar)          self.timer.start(1000)  # Update every second          mainLayout = QVBoxLayout()          vbox1 = QVBoxLayout()          vbox2 = QVBoxLayout()          vbox3 = QVBoxLayout()          hbox1 = QHBoxLayout()            #Layout - attribute          altitude\_lay = self.setAltitudeGraph()          a2\_lay = self.seta2Graph()          a3\_lay = self.seta3Graph()          visual3D\_lay = self.Visual3DMotion()          groupbutton\_lay = self.setgroupButton()          vbox1.addLayout(altitude\_lay,2)          vbox1.addLayout(a2\_lay,2)          vbox1.addLayout(a3\_lay,2)          vbox2.addLayout(visual3D\_lay,4)          vbox2.addStretch(3)          vbox3.addStretch(4)          vbox3.addLayout(groupbutton\_lay,2)          hbox1.addLayout(vbox1, 3)          hbox1.addLayout(vbox2, 4)          hbox1.addLayout(vbox3, 2)          mainLayout.addLayout(hbox1)          # Create a central widget and set the layout          centralWidget = QWidget()          centralWidget.setLayout(mainLayout)          self.setCentralWidget(centralWidget)          #self.setLayout(vbox)          self.show()      #상태창      def updateStatusBar(self):          current\_date\_time = QDateTime.currentDateTime().toString('yyyy-MM-dd HH:mm:ss')          self.statusBar().showMessage(current\_date\_time)      #고도 그래프      def setAltitudeGraph(self):          self.altitude = graph\_altitude()          vbox = QHBoxLayout()          vbox.addWidget(self.altitude)          return vbox        #3D 시각화 로켓 자세      def Visual3DMotion(self):          self.visual3D = visual3D()            vbox = QHBoxLayout()          vbox.addWidget(self.visual3D)          return vbox        # groupbutton      def setgroupButton(self):          self.groupButton = group\_button()            vbox = QHBoxLayout()          vbox.addWidget(self.groupButton)          return vbox        # 그래프 2      def seta2Graph(self):          self.a2 = graph\_a2()          vbox = QHBoxLayout()          vbox.addWidget(self.a2)          return vbox      # 그래프 3      def seta3Graph(self):          self.a3 = graph\_a3()          vbox = QHBoxLayout()          vbox.addWidget(self.a3)          return vbox      def drawGraphs(self):          random\_number = random.randint(1,10)          self.graph.update(random\_number)  if \_\_name\_\_ == '\_\_main\_\_':      app = QApplication(sys.argv)      ex = guiMain()      sys.exit(app.exec\_()) | | Mpu6050 | | #include <Adafruit\_MPU6050.h>  #include <Adafruit\_Sensor.h>  #include <Wire.h>  Adafruit\_MPU6050 mpu;  unsigned long previousTime = 0;  float roll = 0.0;  float pitch = 0.0;  float yaw = 0.0;  void setup(void) {    Serial.begin(115200);    while (!Serial)      delay(10); // will pause Zero, Leonardo, etc until serial console opens    Serial.println("Adafruit MPU6050 test!");    // Try to initialize!    if (!mpu.begin()) {      Serial.println("Failed to find MPU6050 chip");      while (1) {        delay(10);      }    }    Serial.println("MPU6050 Found!");    mpu.setAccelerometerRange(MPU6050\_RANGE\_8\_G);    mpu.setGyroRange(MPU6050\_RANGE\_500\_DEG);    mpu.setFilterBandwidth(MPU6050\_BAND\_21\_HZ);    Serial.println("");    delay(100);  }  void loop() {    /\* Get new sensor events with the readings \*/    sensors\_event\_t a, g, temp;    mpu.getEvent(&a, &g, &temp);    // Calculate the time elapsed    unsigned long currentTime = millis();    float dt = (currentTime - previousTime) / 1000.0;    previousTime = currentTime;    // Calculate pitch and roll from accelerometer data    float accelPitch = atan2(a.acceleration.y, sqrt(a.acceleration.x \* a.acceleration.x + a.acceleration.z \* a.acceleration.z)) \* 180.0 / PI;    float accelRoll = atan2(-a.acceleration.x, a.acceleration.z) \* 180.0 / PI;    // Gyroscope integration for pitch, roll, and yaw    roll += g.gyro.x \* dt;    pitch += g.gyro.y \* dt;    yaw += g.gyro.z \* dt;    // Complementary filter to combine accelerometer and gyroscope data    float alpha = 0.98; // Weight for gyroscope data    roll = alpha \* roll + (1 - alpha) \* accelRoll;    pitch = alpha \* pitch + (1 - alpha) \* accelPitch;    // /\* Print out the values \*/    // Serial.print(pitch);    // Serial.print(",");    // Serial.print(roll);    // Serial.print(",");    // Serial.println(yaw);    // //Serial.println(g.gyro.z);    Serial.print(g.gyro.x);    Serial.print(",");    Serial.print(g.gyro.y);    Serial.print(",");    Serial.println(g.gyro.z);    } | |
| **활동 결과** | *※ 활동 내용을 통해 나온 결과 정리*  GUI에서 맡을 역할을 분담하고, 무엇을 사용할 계획(PyQt)를 정하고, 어떻게 구현할 것인지 정리하였다.      이전에 말했던 딜레이는 아두이노 코드 상에 딜레이 값을 많이 주어 생긴 것이다. 지금 코드는 각속도를 이용해 돌리는 것인데 정확한 자세를 위해서는 mpu6050의 자이로와 가속도 값을 이용해 roll, pitch, yaw 방향으로 틀어진 각도를 추정해서 나타내야한다. 이부분은 지금 고민중이다. |
| **계획** | *※ 다음 활동 시 해야할 일*  위에 내용을 좀 더 구체화 할 것이다. Pygame 을 다른 버튼과 패널 등과 함께 보이게 할것이고 , yaw 부분의 값 누적 오차에 대해서 좀 더 생각해보야할 것 같다. |

2024년 5 월 19 일

작성자 : 김민주, 장현주 (인)