Develop an python code

TEAM ID	PNT2022TMID46814
PROJECT NAME	Child safety monitoring and notification system

Temperature and Humidity sensor code:

```
"""
'temp_humidity.py'
==================================
Example of sending analog sensor
values to an Adafruit IO feed.

Author(s): Brent Rubell

Tutorial Link: Tutorial Link: https://learn.adafruit.com/adafruit-io-basics-temperature-and-humidity

Dependencies:
   - Adafruit IO Python Client
      (https://github.com/adafruit/io-client-python)
   - Adafruit_Python_DHT
      (https://github.com/adafruit/Adafruit_Python_DHT)
"""

# import standard python modules.
import time

# import adafruit dht library.
import Adafruit_DHT

# import Adafruit IO REST client.
from Adafruit_IO import Client, Feed
```

```
# Delay in-between sensor readings, in seconds.
DHT_READ_TIMEOUT = 5

# Pin connected to DHT22 data pin
DHT_DATA_PIN = 26

# Set to your Adafruit IO key.
# Remember, your key is a secret,
# so make sure not to publish it when you publish this code!
ADAFRUIT_IO_KEY = 'YOUR_AIO_KEY'

# Set to your Adafruit IO username.
# (go to https://accounts.adafruit.com to find your username).
ADAFRUIT_IO_USERNAME = 'YOUR_AIO_USERNAME'

# Create an instance of the REST client.
aio = Client(ADAFRUIT_IO_USERNAME, ADAFRUIT_IO_KEY)

# Set up Adafruit IO Feeds.
temperature_feed = aio.feeds('temperature')
humidity_feed = aio.feeds('humidity')

# Set up DHT22 Sensor.
dht22_sensor = Adafruit_DHT.DHT22

while True:
    humidity, temperature = Adafruit_DHT.read_retry(dht22_sensor, DHT_DATA_PIN)
    if humidity is not None and temperature is not None:
        print('Temp={0:0.1f}*C Humidity={1:0.1f}%'.format(temperature, humidity))
        # Send humidity and temperature feeds to Adafruit IO
        temperature = '%.2f'%(temperature)
        humidity = '%.2f'%(humidity)
        aio.send(temperature_feed.key, str(temperature))
        aio.send(humidity_feed.key, str(humidity))
    else:
        print('Failed to get DHT22 Reading, trying again in ', DHT_READ_TIMEOUT,
'seconds')
    # Timeout to avoid flooding Adafruit IO
    time.sleep(DHT_READ_TIMEOUT)
```

Panic button specs setup:

```
#
# spec file for package Panic
#
# Copyright (c) 2016 SUSE LINUX Products GmbH, Nuernberg, Germany.
#
# All modifications and additions to the file contributed by third parties
# remain the property of their copyright owners, unless otherwise agreed
# upon. The license for this file, and modifications and additions to the
# file, is the same license as for the pristine package itself (unless the
# license for the pristine package is not an Open Source License, in which
# case the license is the MIT License). An "Open Source License" is a
# license that conforms to the Open Source Definition (Version 1.9)
# published by the Open Source Initiative.

# Please submit bugfixes or comments via http://bugs.opensuse.org/
#

%define _modname panic

Name:           Panic
Version:        5.5
Release:        0
License:        GPL-3.0+
Summary:        Alarm System toolkit
Url:            http://www.tango-controls.org/community/projects/panic/
Group:          Productivity/Scientific/Other
Source:         %{_modname}-%{version}.tar.gz
BuildRoot:      %{_tmppath}/%{name}-%{version}-build
BuildArch:      noarch

BuildRequires:  python-devel
BuildRequires:  python-setuptools

Requires: python < 3
Requires: python-numpy
```

```
Requires: python-taurus
Requires: python-pytango
Requires: python-fandango

%description
PANIC Alarm System is a set of tools (api, Tango device server, user interface) that provides:
    Periodic evaluation of a set of conditions.
    Notification (email, sms, pop-up, speakers)
    Keep a log of what happened. (files, Tango Snapshots)
    Taking automated actions (Tango commands / attributes)
    Tools for configuration/visualization.


# ======================================================================
# python-panic
# ======================================================================

%package -n python-panic
Summary:                Alarm System toolkit. Python module
Group:      Development/Languages/Python

%description -n python-panic
This package provides the "panic" python module from the PANIC Alarm System toolkit

# ======================================================================
# tangods-pyalarm
# ======================================================================

%package -n tangods-pyalarm
Summary:        Alarm System toolkit. PyAlarm Tango Device Server
Group:          Development/Libraries
Requires:       python-panic
#TODO: we should find a better group for DSs

%description -n tangods-pyalarm
This package provides the PyAlarm Tango Device Server from the PANIC Alarm System toolkit




%prep
%setup -n %{_modname}-%{version}

%build



%install
python setup.py install --prefix=%{_prefix} --root=%{buildroot}



%files -n python-panic
%defattr(-,root,root)
%exclude %{python_sitelib}/%{_modname}/ds
%{python_sitelib}
```

```
%files -n tangods-pyalarm
%defattr(-,root,root)
%{python_sitelib}/%{_modname}/ds
%{_bindir}/*
```

Panic button code:

```
#!/bin/sh

PANIC=$(python -c "import imp;print(imp.find_module('panic')[1])")

#PANIC_DEFAULT="--filter=!building"

HASSCREEN=$(which screen 2>/dev/null)
CMD="python $PANIC/gui/gui.py $PANIC_DEFAULT"

if [ "$( echo $1 | grep '\?\|\-h' )" ] ; then
  echo "Usage:"
  echo "    > panic [-?] [-v/--attach] [--filter=...]"
  echo ""
  echo "Console output will be disabled unless attach/v options are passed"
  echo "SCREEN is used as default shell if present"
  echo ""

else
  echo "Launching panic-gui ..."
  if [ "$*" ] && [ "$( echo "$*" | grep '\-v' )" ] ; then
    echo "raw"
    CMD=${CMD}

  elif [ "$*" ] && [ "$( echo "$*" | grep 'attach' )" ] ; then
    echo "screen"
    if [ "${HASSCREEN}" ] ; then
      CMD="screen -S panic-gui ${CMD}"
    fi

  else
    echo "run detached"
    if [ "${HASSCREEN}" ] ; then
      CMD="screen -dm -S panic-gui ${CMD}"
    else
      CMD="${CMD} > /dev/null"
    fi
  fi
  echo $CMD $*
  $CMD $*
fi
```

Pressure sensor:

```
import serial, time

import RPi.GPIO as GPIO
import re

import smtplib,ssl

from email.mime.multipart import MIMEMultipart

from email.mime.base import MIMEBase

from email.mime.text import MIMEText

from email.utils import formatdate

from email import encoders


bp = 2 // Bp sensor status pin

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)

GPIO.setup(bp,GPIO.IN)

ser = serial.Serial("/dev/ttyS0",9600)

ser.bytesize = serial.EIGHTBITS #number of bits per bytes

ser.parity = serial.PARITY_NONE #set parity check: no parity

ser.stopbits = serial.STOPBITS_ONE #number of stop bits

ser.timeout = 1           #non-block read

ser.xonxoff = False     #disable software flow control

ser.rtscts = False     #disable hardware (RTS/CTS) flow control

ser.dsrdtr = False       #disable hardware (DSR/DTR) flow control

file = open('sensor_readings.txt', 'w')

#file.write('time and date, systolic(b), diastolic(c),Pulse(d)\n')

file.write(' Your Blood Pressure Details Are As Follows\n')



def send_an_email_normal():

   toaddr = 'to@gmail.com'      # To id

   me = 'your email id'         # your id

   subject = "Normal Blood pressure"            # Subject

   msg = MIMEMultipart()

   msg['Subject'] = subject
   msg['From'] = me
   msg['To'] = toaddr
   msg.preamble = "test "
   #msg.attach(MIMEText(text))

   part = MIMEBase('application', "octet-stream")
   part.set_payload(open("sensor_readings.txt", "rb").read())
   encoders.encode_base64(part)
   part.add_header('Content-Disposition', 'attachment; filename="sensor_readings.txt"')   # File
name and format name
   msg.attach(part)

   try:
     s = smtplib.SMTP('smtp.gmail.com', 587)  # Protocol
     s.ehlo()
     s.starttls()
     s.ehlo()
     s.login(user = 'mailto:xxxxx@gmail.com', password = 'your email password')  # User id &
password
     #s.send_message(msg)
     s.sendmail(me, toaddr, msg.as_string())
     s.quit()
   #except:
   #   print ("Error: unable to send email")
   except SMTPException as error:
        print ("Error")              # Exception

def send_an_email_Hypotension():
   toaddr = 'to@gmail.com'      # To id
   me = 'mailto:your email id'         # your id
   subject = "Hypotension"             # Subject
   msg = MIMEMultipart()
   msg['Subject'] = subject

   msg['From'] = me

   msg['To'] = toaddr

   msg.preamble = "test "

   #msg.attach(MIMEText(text))


   part = MIMEBase('application', "octet-stream")

   part.set_payload(open("sensor_readings.txt", "rb").read())

   encoders.encode_base64(part)

   part.add_header('Content-Disposition', 'attachment; filename="sensor_readings.txt"')   # File
name and format name

   msg.attach(part)


   try:

     s = smtplib.SMTP('smtp.gmail.com', 587)  # Protocol

     s.ehlo()

     s.starttls()

     s.ehlo()

     s.login(user = 'mailto:xxxxx@gmail.com', password = 'your email password') # User id &
password

     #s.send_message(msg)

     s.sendmail(me, toaddr, msg.as_string())

     s.quit()

   #except:

   #   print ("Error: unable to send email")

   except SMTPException as error:

       print ("Error")              # Exception


def send_an_email_prehypertension():

   toaddr = 'to@gmail.com'      # To id

   me = 'mailto:your email id'         # your id

   subject = "Prehypertension"            # Subject
```

```
    msg = MIMEMultipart()
    msg['Subject'] = subject
    msg['From'] = me
    msg['To'] = toaddr
    msg.preamble = "test "
    #msg.attach(MIMEText(text))

    part = MIMEBase('application', "octet-stream")
    part.set_payload(open("sensor_readings.txt", "rb").read())
    encoders.encode_base64(part)
    part.add_header('Content-Disposition', 'attachment; filename="sensor_readings.txt"')   # File
name and format name
    msg.attach(part)

    try:
       s = smtplib.SMTP('smtp.gmail.com', 587)  # Protocol
       s.ehlo()
       s.starttls()
       s.ehlo()
       s.login(user = 'mailto:xxxxx@gmail.com', password = 'your email password') # User id &
password
       #s.send_message(msg)
       s.sendmail(me, toaddr, msg.as_string())
       s.quit()
    #except:
    #   print ("Error: unable to send email")
    except SMTPException as error:
          print ("Error")              # Exception

def send_an_email_stage1():
    toaddr = 'to@gmail.com'      # To id
    me = 'mailto:your email id'         # your id
```

```
   subject = "Stage 1 Hypertension"            # Subject
   msg = MIMEMultipart()
   msg['Subject'] = subject
   msg['From'] = me
   msg['To'] = toaddr
   msg.preamble = "test "
   #msg.attach(MIMEText(text))

   part = MIMEBase('application', "octet-stream")
   part.set_payload(open("sensor_readings.txt", "rb").read())
   encoders.encode_base64(part)
   part.add_header('Content-Disposition', 'attachment; filename="sensor_readings.txt"')   # File
name and format name
   msg.attach(part)

   try:
     s = smtplib.SMTP('smtp.gmail.com', 587)  # Protocol
     s.ehlo()
     s.starttls()
     s.ehlo()
      s.login(user = 'mailto:xxxxx@gmail.com', password = 'your email password') # User id &
password
     #s.send_message(msg)
     s.sendmail(me, toaddr, msg.as_string())
     s.quit()
   #except:
   #   print ("Error: unable to send email")
   except SMTPException as error:
       print ("Error")              # Exception

def send_an_email_stage2():
   toaddr = 'to@gmail.com'      # To id
   me = 'mailto:your email id'         # your id
   subject = "Stage 2 Hypertension"            # Subject
   msg = MIMEMultipart()
   msg['Subject'] = subject
   msg['From'] = me
   msg['To'] = toaddr
   msg.preamble = "test "
   #msg.attach(MIMEText(text))

   part = MIMEBase('application', "octet-stream")
   part.set_payload(open("sensor_readings.txt", "rb").read())
   encoders.encode_base64(part)
   part.add_header('Content-Disposition', 'attachment; filename="sensor_readings.txt"')   # File
name and format name
   msg.attach(part)

   try:
      s = smtplib.SMTP('smtp.gmail.com', 587)  # Protocol
      s.ehlo()
      s.starttls()
      s.ehlo()
      s.login(user = 'mailto:xxxxx@gmail.com', password = 'your email password') # User id &
password
      #s.send_message(msg)
      s.sendmail(me, toaddr, msg.as_string())
      s.quit()
   #except:
   #   print ("Error: unable to send email")
   except SMTPException as error:
        print ("Error")              # Exception

def send_an_email_hypertensive():
   toaddr = 'to@gmail.com'     # To id
   me = 'mailto:your gmail id'         # your id
   subject = "Hypertensive Crisis"            # Subject
   msg = MIMEMultipart()
   msg['Subject'] = subject
   msg['From'] = me
   msg['To'] = toaddr
   msg.preamble = "test "
   #msg.attach(MIMEText(text))

   part = MIMEBase('application', "octet-stream")
   part.set_payload(open("sensor_readings.txt", "rb").read())
   encoders.encode_base64(part)
   part.add_header('Content-Disposition', 'attachment; filename="sensor_readings.txt"')   # File
name and format name
   msg.attach(part)

   try:
      s = smtplib.SMTP('smtp.gmail.com', 587)  # Protocol
      s.ehlo()
      s.starttls()
      s.ehlo()
      s.login(user = 'mailto:xxxxx@gmail.com', password = 'your email password') # User id &
password
      #s.send_message(msg)
      s.sendmail(me, toaddr, msg.as_string())
      s.quit()
   #except:
   #   print ("Error: unable to send email")
   except SMTPException as error:
        print ("Error")             # Exception
```

```
while True:
  response = ser.readline().decode('ASCII')
  x = len(response)# checking the avalible bytes in the serial data
  if x >= 10:
      print("systolic,diastolic,Pulse")
      print(response)
      y = response.split(',') #spliting the each bytes with comma
      print(y)
      z = re.findall('[0-9]+', response) # extracting each byte
      print(z)
      a = [z] # creating extracted each byte in a array
      b = int(z[0]) # first byte in a array (Systollic)
      c = int(z[1]) # second byte in a array (diastollic)
      d = int(z[2]) # third byte in a array (pulse)
      print("Systolic :", b)
      print("Diastollic :",c)
      print("Pulse:" , d)
      file.write(time.strftime('%H:%M:%S %d/%m/%Y') + '   Systollic:' + str(b) + '  Diastollic:'+ str(c)+'
Pulse rate:' + str(d) + '\n')
      time.sleep(1)
      file.close()
      print("file written")
      if b < 90 and c < 60:
          print('Hypotension')
          send_an_email_Hypotension()


      if b > 120 and b < 139 and c > 80 and c < 90:
          print('Prehypertension')
          send_an_email_prehypertension()


      if b > 140 and b < 159 and c > 90 and c < 99:
```

```
    print('stage 1 Hypertension')
    send_an_email_stage1()

if b > 160 and b < 179 and c > 100 and c < 109:
    print('stage 2 Hypertension')
    send_an_email_stage2()

if b > 180 and c > 110:
    print('Hypertensive Crisis')
    send_an_email_hypertensive()

else:
    send_an_email_normal()
    print('sent_the_mail')
```