Project Development

Delivery of sprint-2

Date	04 november2022
Team ID	PNT2022TMID49530
Project Name	IOT Based Smart Crop Protection for Agriculture
Maximum Marks	2 Marks

Sprint-2 coding

Detect The PH Level of Crops

```
import io  # used to create file streams
import fcntl  # used to access I2C parameters like addresses

import time  # used for sleep delay and timestamps


class Ezo:
   long_timeout = 1.5  # the timeout needed to query readings and
                  #calibrations
   short_timeout = .5  # timeout for regular commands
   default_bus = 1  # the default bus for I2C on the newer Raspberry Pis,
               # certain older boards use bus 0
   default_address = 99  # the default address for the pH sensor

   def __init__(self, address=default_address, bus=default_bus):
      # open two file streams, one for reading and one for writing
      # the specific I2C channel is selected with bus
      # it is usually 1, except for older revisions where its 0
      # wb and rb indicate binary read and write
      self.file_read = io.open("/dev/i2c-" + str(bus), "rb", buffering=0)
      self.file_write = io.open("/dev/i2c-" + str(bus), "wb", buffering=0)

      # initializes I2C to either a user specified or default address
      self.set_i2c_address(address)

   def set_i2c_address(self, addr):
      # set the I2C communications to the slave specified by the address
      # The commands for I2C dev using the ioctl functions are specified in
      # the i2c-dev.h file from i2c-tools
      I2C_SLAVE = 0x703
      fcntl.ioctl(self.file_read, I2C_SLAVE, addr)
      fcntl.ioctl(self.file_write, I2C_SLAVE, addr)

   def write(self, string):
      # appends the null character and sends the string over I2C
      string += "\00"
      self.file_write.write(bytes(string, 'UTF-8'))

   def read(self, num_of_bytes=31):
      # reads a specified number of bytes from I2C,
      # then parses and displays the result
      res = self.file_read.read(num_of_bytes)  # read from the board
      # remove the null characters to get the response
      response = [x for x in res if x != '\x00']
      if response[0] == 1:  # if the response isnt an error
          # change MSB to 0 for all received characters except the first
          # and get a list of characters
          char_list = [chr(x & ~0x80) for x in list(response[1:])]
          # NOTE: having to change the MSB to 0 is a glitch in the
          # raspberry pi, and you shouldn't have to do this!
          # convert the char list to a string and returns it
          #return "Command succeeded " +
          return''.join(char_list)
      else:
          return "Error " + str(response[0])

   def query(self, string):
      # write a command to the board, wait the correct timeout,
      # and read the response
      self.write(string)

      # the read and calibration commands require a longer timeout
      if((string.upper().startswith("R")) or
         (string.upper().startswith("CAL"))):
          time.sleep(self.long_timeout)
      elif((string.upper().startswith("SLEEP"))):
          return "sleep mode"
      else:
          time.sleep(self.short_timeout)

      return self.read()

   def close(self):
      self.file_read.close()
      self.file_write.close()

#ph = Ezo()
#phvalue = ph.query('R')
#ph1 = str(phvalue)
#ph2 = round(phvalue)
#print (ph.query('R'))
#print (round(ph.query('R'),2))
```