**Cybernetic system for Environmental Monitoring**

*Sasha Fathima Suhel*

*AUD 9521*

*sashaS@amitydubai.ae*

**ABSTRACT**

This paper presents the improvement of a cyber-physical framework which screens the ecological states alternately the encompassing states for example, temperature, pressure in indoor spaces in remote areas. The correspondences between Transmitters and Receivers are performed towards remote foundations. The resulted solution gives likelihood about logging estimations from areas throughout the world and for visualizing and examining this assembled information from the device associated with Internet. The whole solution comprises this effort, Cyber-physical system that consists of requested sensors, processor and communications protocol starting at the physical level, and which achieves information management and storage at the digital level. The test results show that the proposed framework is a viable and easy option for economic monitoring applications.

**Keywords:** Cyber-physical system, Internet of things; Wireless communication, wireless infrastructure, sensors, communication protocol

**INTRODUCTION**

The requirements for environment care are speedily rising with the continuous growing of the world population. Incredible success in electronic technology has been observed with the rapid advancement of sensors, Raspberry Pi and computers prompt development. Numerous technological revolutionary designs are taken for the advantage of electronic service improvement. By using sensors, the data is gathered. After gathering certain information, the data and status of each sensor are updated through IOT. This data is then used to analyse the data and tell us the best possible time frame to cultivate crops at our garden home.

Agriculture in India is based on the rainfall situation such as summer monsoon, winter monsoon etc. Crops are harvested according to those seasons but due to the vast climate change there is no standard at which the crops must be harvested. Due to this problem farmers follow their manual data that is provided to them and cannot follow a specific time frame or depend on the rain.

For crop farmers, timing is important since the payoff from your produce is around the corner. The date of the harvest is governed mostly by humidity. However, for each farm the time of harvest is variable, dependent on grain management capacity of the farm. Certain farms harvest 32-34 percent of high humidity crops. Subsequent drying processes might commence at a moisture content of 28 percent to dry the grain before storage. In the field, many farmers have allowed the grain to dry out and harvest at 15% or 16% moisture.

Household gardeners do not have access to this kind of manual data to grow plants at their home, as there are so many different environmental factors that play a role in finding the best time to pluck the plant. Hence why most people at home while gardening pluck their crops either just unripe or too ripe and find it either confusing or difficult to get the timing just right. To help home gardeners to get the best possible outcome its necessary to procure discrete values to enhance productivity.

Timely and precise monitoring can improve crop management, harvest planning and processing. Differential equations are used to describe and forecast discrete events. They are widely used in numerous domains, including economics, quantum physics, environmental sciences and meteorology. Many biological events are represented using differential equations and can be employed in costly or unethical research. The models were utilized to provide ways to achieve the desired condition.

**METHODOLOGY**

There are 3 different sensors connected to the Raspberry pi. The sensors being DHT11, Ultrasonic Distance Sensor (HC-SR04) and. The hardware takes in the input of the temperature and humidity and sends it back to the monitoring system. The ultrasonic sensor is connected to the led/buzzer which then signals the home user when there is any obstacle closer than 20cm.

The software runs on real time and takes in the temperature and humidity in 4 different time intervals throughout the day. Each interval will be taken in large numbers of data values at a time. A GUI is created to display current readings as well as save current 100 readings at that given point in time.

It also shows a prediction map for 2 crops which could be set to a n number of crops or plants a user might have at their home, by knowing the ideal temperature to humidity ratio of the given crop, the user can pluck the crop out at the best possible temperature and humidity. This is done to increase best possible outcome of the crop.

**BLOCK DIAGRAM**

Diagram, schematic

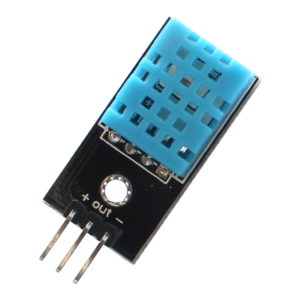
Description automatically generated

**COMPONENTS**

**DHT11**

[1] The DHT11 is a frequently used sensor of moisture and temperature. The sensor includes an NTC for the temperature measurement and an 8-bit microprocessor for serial data output of temperature and humidity information. The sensor is fitted and hence easy to connect with various microcontrollers.

The sensor can measure temperature from 0°C to 50°C and humidity from 20% to 90% with an accuracy of ±1°C and ±1%.



**DHT11 Adafruit Library**

Python library to read the DHT series of humidity and temperature sensors on a Raspberry Pi or Beaglebone Black.

Designed specifically to work with the Adafruit DHT series sensors

**Ultrasonic Distance Sensor (HC-SR04)**

The ultrasonic sensor (or transducer) uses the same radar system concepts. An ultrasound sensor may transform electrical energy into acoustic waves. The acoustic wave signal is an ultrasound wave that is above 18kHz in frequency. The renowned HC SR04 ultrasonic sensor produces 40kHz of ultrasonic waves.

Usually, for communication with ultrasonic sensors a microcontroller is employed. The microprocessor sends a trigger signal to the ultrasonic sensor to start measuring the distance. For the HC-SR04 ultrasound sensor the duty cycle of this trigger signal is 10μS. The ultrasound sensor creates a time clock when it is activated eight (ultrasonic) waves explode. The timer will terminate whenever the mirrored signal (echo) is received. The output of the ultrasound sensor is a strong pulse with the same duration as the time difference between echo signal and transmitted ultrasound explosions.

**Raspberry Pi**

[3] The Raspberry Pi is a very cheap computer that runs Linux, but it also provides a set of GPIO (general purpose input/output) pins, allowing you to control electronic components for physical computing and explore the Internet of Things.

**Python Compiler**

Python is a high-level language for generic programming. With its remarkable usage of extensive indentation Python's design philosophy prioritizes code reading. Its language and its object-oriented approach assist programmers to develop clear, logical codes for large and small projects.

**Bread Board or Universal Soldering Board**

[4] A breadboard is a solder-free plug-and-play platform for (usually) fast insertion and removal of electrical components into circuit-construction applications in today's electronics and engineering circles.

Today, the breadboard kits are a popular toolbox product for professionals, amateurs and enthusiasts:

A picture containing text, electronics

Description automatically generated

**Jumper Wires**

[5] Two dots on the circuit are connected by jumper wires. In a range of lengths and variants all electronic inventories jump wire. It is often used with breadboards and other prototyping equipment to facilitate changing a circuit straightforward if necessary. Male jumpers are designed to plug securely into the holes in a breadboard. Female jumpers are useful for connecting male header posts and pin terminals on components. Jumpers are available in female-female, male-male and male-female configurations.

**Python Special Libraries**

[6] While the Python Language Reference defines the Python language's actual syntax and semantics, that handbook explains Python's standard library. It also discusses some of the optional components usually included in distributions of Python.

The standard library of Python is quite widespread and offers a large number of capabilities, as seen in the long table of contents below. The library consists of built-in (written in C) modules, which provide access to system features like I/O file which Python programmers would otherwise not access, and modules written in Python, which give standardized solutions for many problems that arise in daily programming. Some of these modules have been expressly created to boost Python's portability by abstracting platform characteristics to platform-neutral APIs.

**Micro SD Card and SD card Reader**

[7] SD (Secure Digital) memory card is manufactured as a novel storage technology based on semi-conductor flash memory to boost digital device capacity. And because of its good properties, compact size, quick data transmission and hot swap, it's liked and utilized by many people.

**Raspbian OS Image and Imager Software**

[8] Raspberry Pi has built an OS, Ubuntu 18.04 and Windows visual SD card writing tool, making it the easiest option for most users when downloading images and installing them onto the SD card.

• Download and install the most recent version of Raspberry Pi Imager.

O You may install it through a terminal with rpi-imager sudo apt, if you prefer to utilize the Raspberry Pi Imager itself.

• Connect an internal SD card reader to the SD card.

• Open Raspberry Pi Imager and select from the list shown the necessary OS.

• Select the SD card you want your image to write to.

Check your choices and start writing data to the SD card by clicking 'WRITE.'

**CODING PROCESS**

from PyQt5 import QtCore, QtGui, QtWidgets

from Sensor\_Node import dhtTopics

from Data\_Schema\_Node import Schema\_Topics

from Prediction\_Node import Prediction\_Topics

class CRGS(object):

def crgs(self, Dialog):

Dialog.setObjectName("Dialog")

Dialog.resize(554, 110)

icon = QtGui.QIcon()

icon.addPixmap(QtGui.QPixmap("Images/CRGS.jpg"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

Dialog.setWindowIcon(icon)

self.label\_2 = QtWidgets.QLabel(Dialog)

self.label\_2.setGeometry(QtCore.QRect(290, 70, 91, 20))

self.label\_2.setObjectName("label\_2")

self.lineEdit = QtWidgets.QLineEdit(Dialog)

self.lineEdit.setGeometry(QtCore.QRect(400, 70, 113, 20))

self.lineEdit.setObjectName("lineEdit")

self.label\_3 = QtWidgets.QLabel(Dialog)

self.label\_3.setGeometry(QtCore.QRect(290, 40, 91, 20))

self.label\_3.setObjectName("label\_3")

self.lineEdit\_2 = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_2.setGeometry(QtCore.QRect(400, 40, 113, 20))

self.lineEdit\_2.setObjectName("lineEdit\_2")

self.pushButton = QtWidgets.QPushButton(Dialog)

self.pushButton.setGeometry(QtCore.QRect(30, 10, 221, 23))

self.pushButton.setObjectName("pushButton")

self.pushButton\_2 = QtWidgets.QPushButton(Dialog)

self.pushButton\_2.setGeometry(QtCore.QRect(30, 40, 221, 23))

self.pushButton\_2.setObjectName("pushButton\_2")

self.pushButton\_3 = QtWidgets.QPushButton(Dialog)

self.pushButton\_3.setGeometry(QtCore.QRect(30, 70, 221, 23))

self.pushButton\_3.setObjectName("pushButton\_3")

self.label\_4 = QtWidgets.QLabel(Dialog)

self.label\_4.setGeometry(QtCore.QRect(400, 10, 81, 16))

self.label\_4.setObjectName("label\_4")

self.pushButton.clicked.connect(self.fetchTemp\_Humid)

self.pushButton\_2.clicked.connect(self.save\_current\_hundred\_Readings)

self.pushButton\_3.clicked.connect(self.get\_Predition)

self.retranslateUi(Dialog)

QtCore.QMetaObject.connectSlotsByName(Dialog)

def retranslateUi(self, Dialog):

\_translate = QtCore.QCoreApplication.translate

Dialog.setWindowTitle(\_translate("Dialog", "Cyber Physical System for Environmental Monitoring and Prediction"))

self.label\_2.setText(\_translate("Dialog", "Temperature"))

self.label\_3.setText(\_translate("Dialog", "Humidity"))

self.pushButton.setText(\_translate("Dialog", "Display Curent Reading"))

self.pushButton\_2.setText(\_translate("Dialog", "Save Current 100 readings"))

self.pushButton\_3.setText(\_translate("Dialog", "Generate Prediction Map"))

self.label\_4.setText(\_translate("Dialog", "12:02:00"))

#System Time Fetcher

self.timer = QtCore.QTimer(Dialog)

self.timer.setInterval(1000)

self.timer.timeout.connect(self.displayTime)

self.timer.start()

def displayTime(self):

self.label\_4.setText(QtCore.QDateTime.currentDateTime().time().toString())

if(str(self.label\_4.text())=="06:00:00"):

print('[autoFetch]')

self.save\_current\_hundred\_Readings()

elif(str(self.label\_4.text())=="12:00:00"):

print('[autoFetch]')

self.save\_current\_hundred\_Readings()

elif(str(self.label\_4.text())=="16:00:00"):

print('[autoFetch]')

self.save\_current\_hundred\_Readings()

elif(str(self.label\_4.text())=="21:00:00"):

print('[autoFetch]')

self.save\_current\_hundred\_Readings()

def fetchTemp\_Humid(self):

dataFetched=dhtTopics.dhtMessages()

self.lineEdit.setText(str(dataFetched[0]))

self.lineEdit\_2.setText(str(dataFetched[1]))

def save\_current\_hundred\_Readings(self):

location='Data\_Schema\_Node\SensorData.csv'

ACK=Schema\_Topics.recordDATA(location)

if(ACK==True):

print('[DATA SAVED]')

def get\_Predition(self):

Prediction\_Topics.Prediction()

if \_\_name\_\_ == "\_\_main\_\_":

import sys

app = QtWidgets.QApplication(sys.argv)

Dialog = QtWidgets.QDialog()

ui = CRGS()

ui.crgs(Dialog)

Dialog.show()

sys.exit(app.exec\_())

**CONCLUSION**

Conclusively, we have developed a cybernetic system for environmental monitoring that essentially helps home gardeners monitor the growth of the plant and the factors affecting it. Three very important elements while home gardeners grow a plant consider are temperature, humidity, and timing. The system is created to generate humidity and temperature data values in real time, for over four times throughout the day. These assist home gardeners grow their plants in a more effective and controlled manner, hence saving time, energy and reducing wastage of resources.

**REFERENCES**

1. <https://components101.com/sensors/dht11-temperature-sensor>
2. <https://pypi.org/project/Adafruit-DHT/>
3. <https://opensource.com/resources/raspberry-pi>
4. <https://uk.rs-online.com/web/generalDisplay.html?id=ideas-and-advice/breadboard-guide>
5. <https://www.allelectronics.com/category/294/jumper-wire/1.html>
6. <https://docs.python.org/3/library/>
7. <https://www.minitool.com/android-recovery/what-is-sd-card-reader-how-to-use-it.html>
8. <https://www.raspberrypi.org/documentation/installation/installing-images/>
9. <https://scholarscompass.vcu.edu/cgi/viewcontent.cgi?article=5557&context=etd>
10. https://www.microcontrollertips.com/principle-applications-limitations-ultrasonic-sensors-faq/