ДОДАТОК А

Копія наукової публікації

Розділ 7

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Андрющенко Д.Д.,
здобувач вищої освіти ступеня магістра
Харківського національного університету радіоелектроніки
науковий керівник: ***Аксак Н.Г.,***
доктор технічних наук,
професор кафедри комп'ютерних інтелектуальних технологій та систем
Харківського національного університету радіоелектроніки

МУЛЬТИАГЕНТНІ СИСТЕМИ ДЛЯ ОБСЛУГОВУВАННЯ ТЕПЛИЧНОГО СЕРЕДОВИЩА

Мультиагентні системи (MAS) це розподілені системи, особливість яких полягає в тому, що їхні компоненти автономні та егоїстичні, які прагнуть задовольнити свої власні цілі. Крім того, ці системи також виділяються тим, що є відкритими системами без централізованої конструкції. Однією з головних причин великого інтересу та уваги до мультигентних систем є те, що вони розглядаються як технологія для складних додатків, які вимагають розподіленої та паралельної обробки даних і працюють автономно в складних і динамічних областях.

Дослідження мультиагентних систем (MAS) базуються на результатах розподілених обчислень, які ставлять нові запитання про те, як агенти повинні взаємодіяти один з одним, щоб координувати свою діяльність і вирішувати складні проблеми [1].

Навчання в MAS є парадигмою великого значення, оскільки система, здатна навчатися і динамічно змінювати свій спосіб дій, має великий потенціал для вирішення багатьох проблем, щодо яких ми не знаємо поведінки інших агентів у навколишньому середовищі.

У мультиагентному середовищі навчання є і важливішим, і складнішим, оскільки вибір дій має здійснюватися в присутності інших агентів, які не обов'язково повинні дотримуватися правил середовища і можуть приймати недетерміновані рішення. Ці агенти, у свою чергу, адаптують свої дії до тих, які раніше виконували інші агенти [2].

Протягом останніх кількох років MAS використовувався як інструмент для розробки багатьох фреймворків AmI(AmbientIntelligence). Як приклад,

можна виділити iGendaframework, основною метою якого є забезпечення інтелектуального керування подіями, що складається з платформи, яка отримує події від інших користувачів і намагається планувати їх відповідно до своєї важливості, маючи можливість створювати, переміщувати та видаляти події [3].

Агентне моделювання — це підхід до моделювання систем, який зосереджується на моделюванні складних технічних систем, які розподілені та передбачають складну взаємодію між людьми та машинами [4]. Моделі імітують одночасні операції кількох сутностей (агентів) у спробі відтворити та передбачати дії складних явищ. Це аварійний процес від самого елементарного рівня (мікро) до найвищого рівня (макро).

Великі досягнення в таких технологіях, як мультиагентні системи, аналітика великих даних, дистанційне зондування, Інтернет речей, системи підтримки прийняття рішень та штучний інтелект, змінили сільськогосподарський сектор. У тепличному секторі ці технології допомагають фермерам підвищити свої прибутки та врожайність, мінімізуючи витрати виробництва, виробляти більш екологічно чистим способом та зменшити ризики, викликані зміною клімату. Зазвичай, виробники, які керують невеликими фермами, у більшості випадків не мають знань, особливо про те, як керувати системами клімат-контролю. Більшість фермерів мають низький рівень освіти [5].

Проте існує багато особливих проблем, включаючи обмежене використання розумних технологій у комерційному сільському господарстві, ціну та точність датчиків. Розвиток технологій з часом сприяв розробці датчиків для запобігання заморозкам, дистанційного моніторингу посівів, запобігання пожежам, точного контролю поживних речовин при вирощуванні теплиць без ґрунту, автономності електроенергії за рахунок використання сонячної енергії та інтелектуального живлення, затінення та освітлення. контроль для підвищення врожайності та зниження експлуатаційних витрат. Також існує багато особливих проблем, включаючи обмежене використання розумних технологій у комерційному сільському господарстві, ціну та точність датчиків [6].

Розроблена система автоматичного поливу кімнатних рослин може бути використана як в побутових умовах, так і довершена та адаптована під розумні теплиці. Система характеризується безперебійним і швидким водопостачанням. Завдяки використанню автоматичного зрошення можна значно знизити трудомісткість і витрати на воду. Концепція дуже проста, але дуже ефективна, тобто весь процес мінімізує втручання людини для забезпечення життєзабезпечення органічних культур та оптимізує всі процеси від вологості ґрунту до температури повітря.

Система автоматичного поливу базується на мікроконтролері Arduino, який керує всіма процесами, датчиками вологості ґрунту та повітря,

датчиками температури повітря та ґрунту, датчиками освітлення, помпою, за допомогою якої відбувається полив, обігрівачами повітря та ґрунту, зволожувачами повітря. Принцип роботи такий – пристрій отримує дані с датчиків, розраховує різницю поточних показників від норми та на основі цього віддає команди всій системі виконувати задані функції. Для розробленого пристрою написаний код, який керує зондом для вимірювання вологи у ґрунті та помпою подачі води. Отримане значення має бути не нижче 50%. Коли вологість падає нижче середнього значення вмикається водяна помпа, яка починає качати воду з резервуару та поливати ґрунт. Процес поливу здійснюється до того моменту, доки вологість ґрунту не досягне середнього рівня.

Впровадження мультиагентної системи дозволяє покращити ефективність планування та контролю. Пропонується використовувати розподілену систему планування кількох агентів на основі кооперативного підходу для вирішення статичних і динамічних задач планування робіт.

Список використаних джерел

1. Baldoni, M., Baroglio, C., May, K. M., Micalizio, R., & Tedeschi, S. (2018). Computational accountability in MAS organizations with ADOPT. *Applied Sciences*, *8*(4), 489.
2. Duan, K., Fong, S., Zhuang, Y., & Song, W. (2018). Artificial neural networks in coordinated control of multiple hovercrafts with unmodeled terms. *Applied Sciences*, *8*(6), 862.
3. Barriuso, A. L., De la Prieta, F., Villarrubia González, G., De La Iglesia, D. H., & Lozano, Á. (2018). MOVICLOUD: Agent-based 3D platform for the labor integration of disabled people. *Applied Sciences*, *8*(3), 337.
4. Miyashita, K. (2017). Incremental design of perishable goods markets through multi-agent simulations. *Applied Sciences*, *7*(12), 1300.
5. Kavga A. et al. Research on innovative training on smart greenhouse technologies for economic and environmental sustainability //Sustainability. – 2021. – Т. 13. – № . 19. – P. 10536.
6. Maraveas C., Bartzanas T. Applicationof Internet of Things (IoT) for Optimized Greenhouse Environments // AgriEngineering. – 2021. – Т. 3. – № 4. – P. 954-970.

Andriushchenko D.D., Aksak N.G.,

Multi-Agent Systems for Service Greenhouse Environment.

The article provides an overview of multi-agent systems and technologies. Multi-agent systems are a subfield of distributed artificial intelligence that is growing rapidly due to the flexibility and intelligence available to solve distributed problems. This chapter provides a brief overview of multi-agent systems. They cover various attributes such as architecture, communication, coordination strategies, decision making, and learning abilities. In the work it is offered to use the distributed system of planning of several agents on the basis of the cooperative approach for the decision of static and dynamic problems of planning of works.

ДОДАТОК Б

Код скетчу Arduino

```
String statusChWriteKey = "?";
id: 1925476
String canalID1 = "?";
String canalID2 = "?";
#include <SoftwareSerial.h>
SoftwareSerialEspSerial(6, 7); // Rx, Tx
// HW Pins
#define FREEZE_LED 13
#define HARDWARE_RESET 8
// DS18B20
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 5 // DS18B20 on pin D5
OneWireoneWire(ONE_WIRE_BUS);
DallasTemperature DS18B20(&oneWire);
97
int soilTemp = 0;
//DHT
#include "DHT.h"
#include <stdlib.h>
int pinoDHT = 11;
int tipoDHT = DHT22;
DHT dht(pinoDHT, tipoDHT);
int airTemp = 0;
int airHum = 0;
// LDR (Light)
#defineldrPIN 1
int light = 0;
// Soil humidity
#definesoilHumPIN 0
int soilHum = 0;

long writeTimingSeconds = 17;
to send data
long readTimingSeconds = 10;
to receive data
long startReadTiming = 0;
long elapsedReadTime = 0;
long startWriteTiming = 0;
long elapsedWriteTime = 0;
//Relays
#define ACTUATOR1 10
#define ACTUATOR2 12
boolean pump = 0;
boolean lamp = 0;
int spare = 0;
```

```
boolean error;
void setup()
{
Serial.begin(9600);
pinMode(ACTUATOR1,OUTPUT);
pinMode(ACTUATOR2,OUTPUT);
pinMode(FREEZE_LED,OUTPUT);
pinMode(HARDWARE_RESET,OUTPUT);
digitalWrite(ACTUATOR1, HIGH);
digitalWrite(ACTUATOR2, HIGH);
digitalWrite(FREEZE_LED, LOW);
digitalWrite(HARDWARE_RESET, HIGH);
 DS18B20.begin();
98
dht.begin();
EspSerial.begin(9600);
EspHardwareReset();
startReadTiming = millis();
startWriteTiming = millis();
}
void loop()
{
 start: //label
 error=0;
elapsedWriteTime = millis()-startWriteTiming;
elapsedReadTime = millis()-startReadTiming;
 if (elapsedReadTime> (readTimingSeconds*1000))
 {
ESPcheck();
 int command = readThingSpeak(canalID1);
 if (command != 9) pump = command;
 delay (5000);
 command = readThingSpeak(canalID2);
 if (command != 9) lamp = command;
takeActions();
startReadTiming = millis();
 }
 if (elapsedWriteTime> (writeTimingSeconds*1000))
 {
ESPcheck();
readSensors();
writeThingSpeak();
startWriteTiming = millis();
 }
 if (error==1)
 {
Serial.println(" <<<< ERROR >>>>");
digitalWrite(FREEZE_LED, HIGH);
 delay (2000);
goto start; //go to label "start"
 }
}
```

```
void readSensors(void)
{
airTemp = dht.readTemperature();
airHum = dht.readHumidity();
99
 DS18B20.requestTemperatures();
soilTemp = DS18B20.getTempCByIndex(0); // Sensor 0 will capture
Soil Temp in Celcius
 light = map(analogRead(ldrPIN), 1023, 0, 0, 100); //LDRDark:0 ==>
light 100%
soilHum = map(analogRead(soilHumPIN), 1023, 0, 0, 100);
}

void takeActions(void)
{
Serial.print("Pump: ");
Serial.println(pump);
Serial.print("Lamp: ");
Serial.println(lamp);
 if (pump == 1) digitalWrite(ACTUATOR1, LOW);
 else digitalWrite(ACTUATOR1, HIGH);
 if (lamp == 1) digitalWrite(ACTUATOR2, LOW);
 else digitalWrite(ACTUATOR2, HIGH);
}

int readThingSpeak(String channelID)
{
startThingSpeakCmd();
 int command;

 String getStr = "GET /channels/";
getStr += channelID;
getStr +="/fields/1/last";
getStr += "\r\n";
 String messageDown = sendThingSpeakGetCmd(getStr);
 if (messageDown[5] == 49)
 {
 command = messageDown[7]-48;
Serial.print("Command received: ");
Serial.println(command);
 }
 else command = 9;
 return command;
}

void writeThingSpeak(void)
{
startThingSpeakCmd();
 String getStr = "GET /update?api_key=";
100
getStr += statusChWriteKey;
```

```
getStr +="&field1=";
getStr += String(pump);
getStr +="&field2=";
getStr += String(window);
getStr +="&field3=";
getStr += String(temp);
getStr +="&field4=";
getStr += String(hum);
getStr +="&field5=";
getStr += String(soilmoisture);
getStr +="&field7=";
getStr += String(illum);
getStr +="&field8=";
getStr += String(spare);
getStr += "\r\n\r\n";
sendThingSpeakGetCmd(getStr);
}
booleanechoFind(String keyword)
{
 byte current_char = 0;
 byte keyword_length = keyword.length();
 long deadline = millis() + 5000; // Tempo de espera 5000ms
 while(millis() < deadline){
 if (EspSerial.available()){
 char ch = EspSerial.read();
Serial.write(ch);
 if (ch == keyword[current_char])
 if (++current_char == keyword_length){
Serial.println();
 return true;
 }
 }
 }
 return false;
}
/********* Reset ESP *************/
void EspHardwareReset(void)
{
Serial.println("Reseting.......");
digitalWrite(HARDWARE_RESET, LOW);
delay(500);
digitalWrite(HARDWARE_RESET, HIGH);
delay(8000);//Tempo necessário para começar a ler
Serial.println("RESET");
}
101

booleanESPcheck(void)
{
EspSerial.println("AT"); // Send "AT+" command to module
 if (echoFind("OK"))
 {
```

```
 //Serial.println("ESP ok");
digitalWrite(FREEZE_LED, LOW);
 return true;
 }
 else //Freeze ou Busy
 {
Serial.println("ESP Freeze
**********************************************************************
***");
digitalWrite(FREEZE_LED, HIGH);
EspHardwareReset();
 return false;
 }
}

void startThingSpeakCmd(void)
{
EspSerial.flush();//limpa o buffer antes de começar a gravar
 String cmd = "AT+CIPSTART=\"TCP\",\"";
cmd += "184.106.153.149"; // Endereco IP de api.thingspeak.com
cmd += "\",80";
EspSerial.println(cmd);
Serial.print("enviado ==> Start cmd: ");
Serial.println(cmd);
if(EspSerial.find("Error"))
 {
Serial.println("AT+CIPSTART error");
 return;
 }
}

String sendThingSpeakGetCmd(String getStr)
{
 String cmd = "AT+CIPSEND=";
cmd += String(getStr.length());
EspSerial.println(cmd);
Serial.print("enviado ==>lenghtcmd: ");
Serial.println(cmd);
if(EspSerial.find((char *)">"))
 {
EspSerial.print(getStr);
102
Serial.print("enviado ==>getStr: ");
Serial.println(getStr);
delay(500);
apresenta busy no próximocomando
 String messageBody = "";
 while (EspSerial.available())
 {
 String line = EspSerial.readStringUntil('\n');
 if (line.length() == 1)
 {
```

```
messageBody = EspSerial.readStringUntil('\n');
 }
 }
Serial.print("MessageBody received: ");
Serial.println(messageBody);
 return messageBody;
 }
 else
 {
EspSerial.println("AT+CIPCLOSE");
Serial.println("ESP8266 CIPSEND ERROR: RESENDING");
 spare = spare + 1;
 error=1;
 return "error";
 }
SoftwareSerialbt(8, 9); // RX, TX

#include <LiquidCrystal.h>
#include "dht.h"
#definedataPin A0
LiquidCrystallcd(2, 3, 4, 5, 6, 7);
dht DHT;

int temp;
int hum;
int soil;
int illum;

void setup() {

Serial.begin(9600);
bt.begin(9600);
Serial.println("Ready");

lcd.begin(16,2);
lcd.setCursor(0,0);
lcd.print(" WELCOME ");
lcd.setCursor(0,1);
lcd.print("Channel");
delay(2000);
lcd.clear();
}

void loop(){
  int readData = DHT.read11(dataPin);

  hum = DHT.humidity;
  temp = DHT.temperature;
  soil = DHT.soilmoisture
  illum = DHT.illuminocity

lcd.setCursor(0,0);
```

```
lcd.print("Humidity: ");
lcd.print(hum);
lcd.print("% ");

lcd.setCursor(0,1);
lcd.print("Temp: ");
lcd.print(temp);
lcd.print((char)223); //degree symbol
lcd.print("C ");

lcd.setCursor(0,2);
lcd.print("Soil moisture: ");
lcd.print(hum);
lcd.print("% ");

lcd.setCursor(0,3);
lcd.print("Illuminocity: ");
lcd.print(hum);
lcd.print("% ");


bt.print(temp); //send distance to MIT App
bt.print(";");
bt.print(hum); //send distance to MIT App
bt.println(";");
bt.print(soil); //send distance to MIT App
bt.println(";");
bt.print(illum); //send distance to MIT App
bt.println(";");

delay(1000);
}
}
```

ДОДАТОК В

Графічна частина магістерської роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

ФАКУЛЬТЕТ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА УПРАВЛІННЯ
КАФЕДРА КІТС

Інтелектуальна система обслуговування тепличного середовища

Магістрант гр. КІТм-21-2 Андрющенко Д.Д.

Науковий керівник проф. Аксак Н.Г.

Харків
2022

Мета роботи

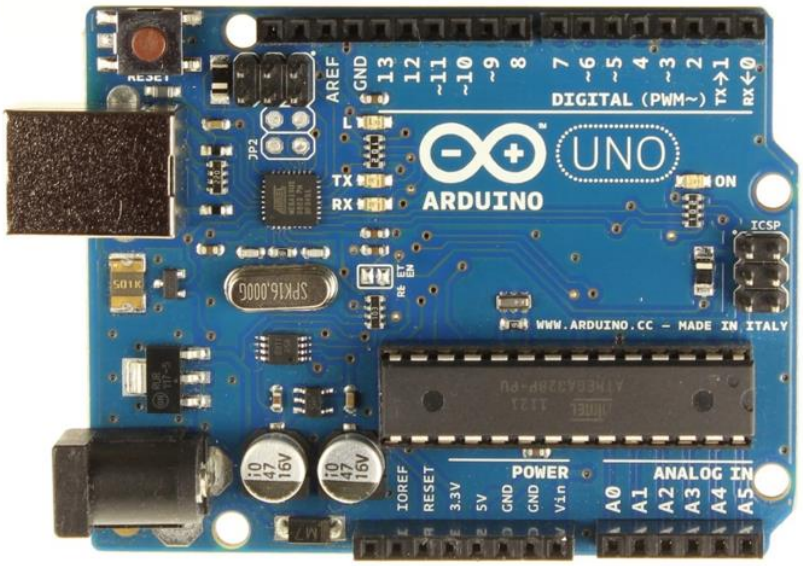
Метою даної роботи є розробка інтелектуальної системи розумної теплиці, здатної працювати в автономному режимі без втручання людини, отримувати та передавати дані на сервер, де вони обробляються та подаються у готовому вигляді у додаток.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- дослідити існуючі інтелектуальні системи обслуговування тепличного середовища та сучасні методи їх створення;
- розробити функціональну схему пристрою забезпечення тепличного середовища ;
- створити програмне забезпечення та мобільний додаток, для інтелектуалізації системи.

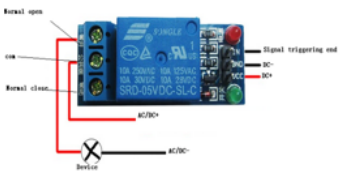
2

Пристрій на базі мікроконтроллера



3

Огляд комплетуючих системи



Реле

Сервопривід

Геркон

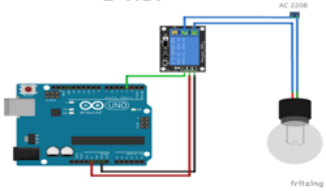


Схема підключення реле

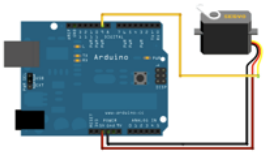


Схема підключення сервоприводу

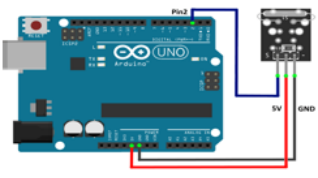


Схема підключення геркону

4

Огляд датчиків системи

Датчик вологості та температури повітря

Датчик рівня води у грунті

Годинник реального часу

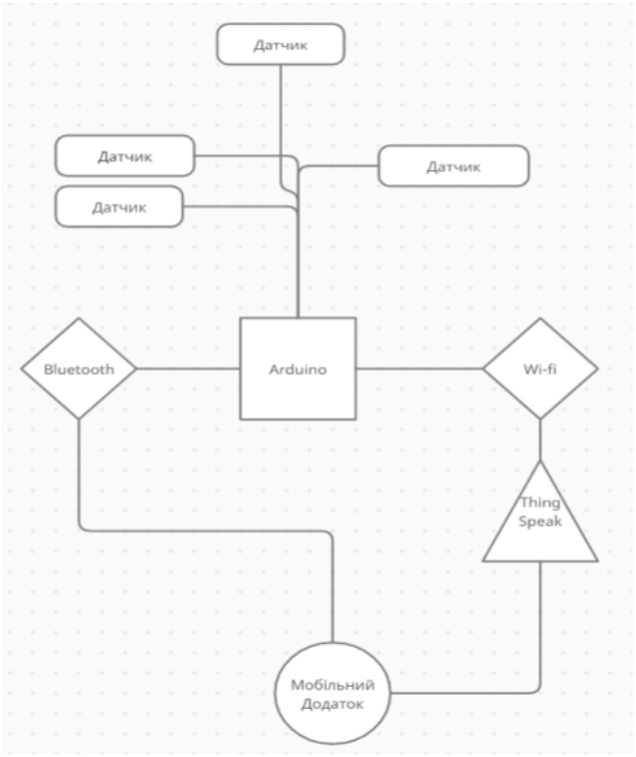
Фото-резистор

Занурювальний датчик

Датчик якості повітря

5

Структурна схема пристрою



6

IoT-платформа Thingspeak

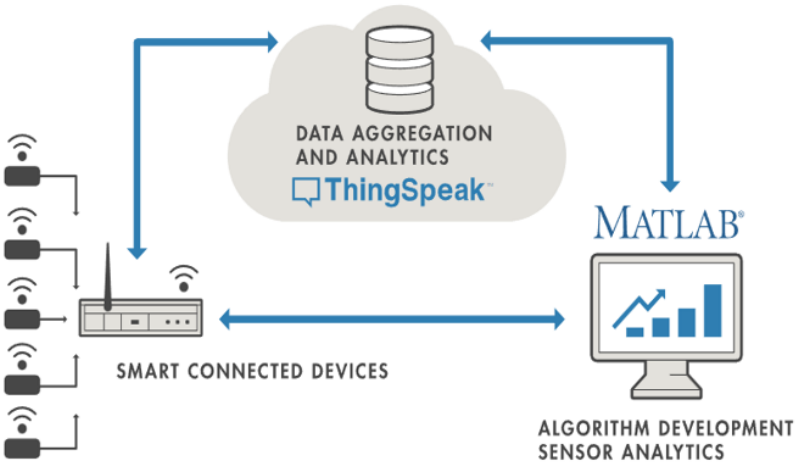
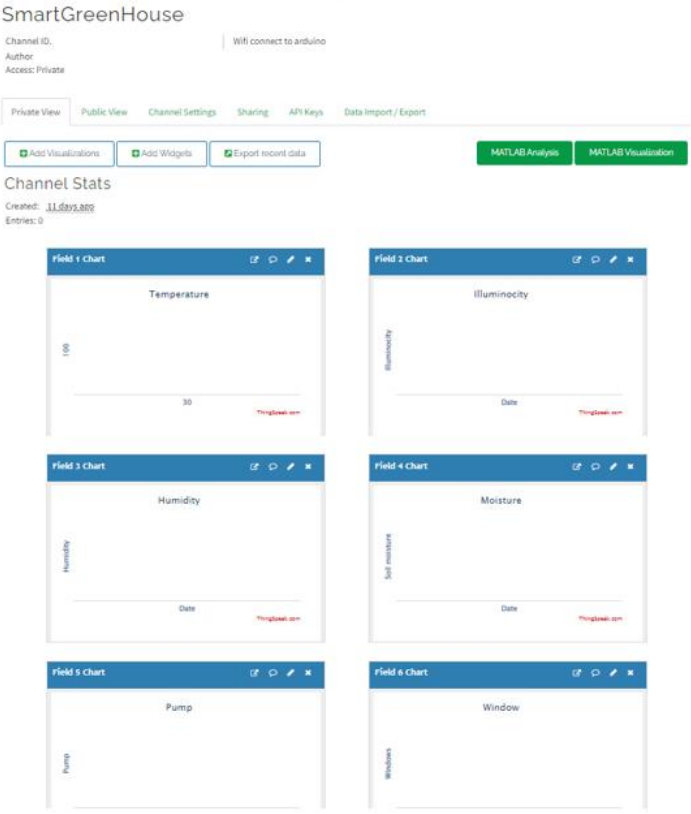


Схема роботи ThingSpeak

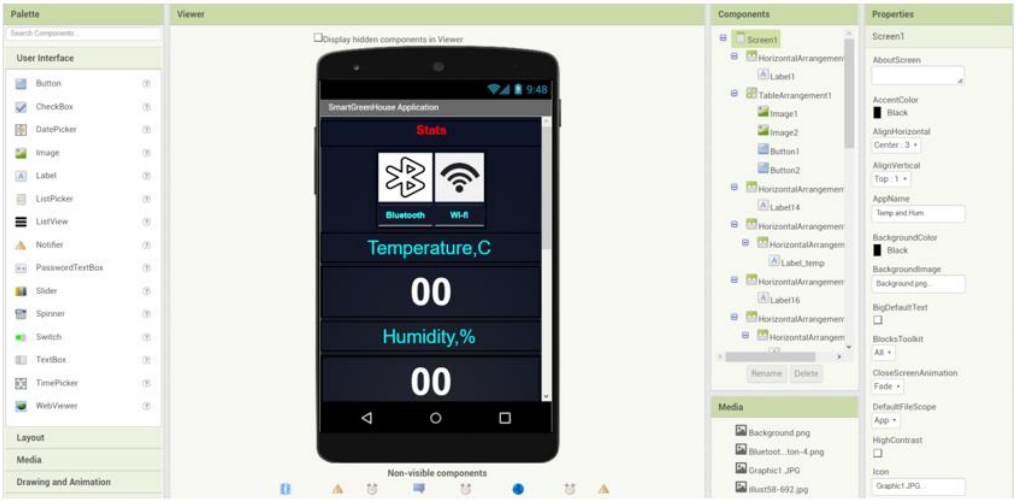
7

Інтерфейс профілю ThingSpeak



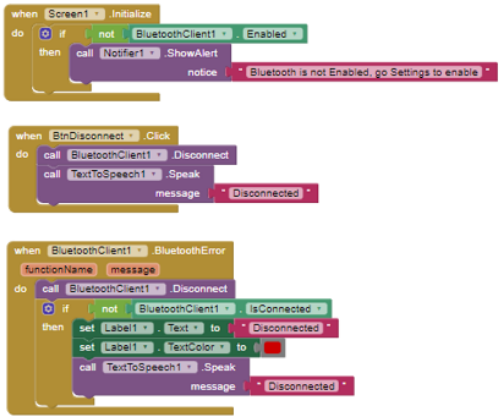
8

Створення інтерфейсу мобільного додатку



Інтерфейс MIT App Inventor

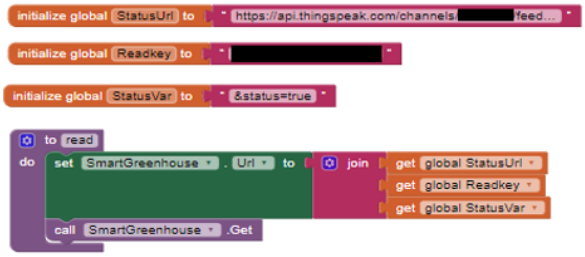
Побудова логіки роботи мобільного додатку через Bluetooth



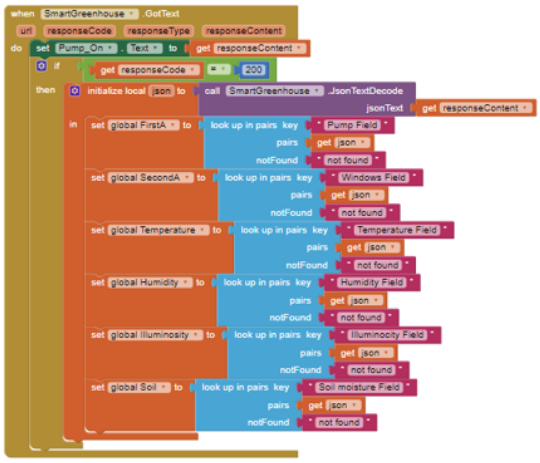
Блок роботи кнопки

Блок передачі даних

Побудова логіки роботи мобільного додатку через Інтернет



Підключення до ThingSpeak



Зчитування JSON-файлів

11

ВИСНОВКИ

- В результаті виконання магістерської роботи було досліджено існуючі інтелектуальні системи обслуговування тепличного середовища, які засновані на багатоагентному підході. Також проаналізовано доцільність використання IoT у таких системах.

- За результатами проведених досліджень було створено систему на основі мікроконтроллера Arduino та датчиків вимірювання показників внутрішнього середовища з використанням багатоагентого підходу, де кожен датчик системи, що підключений до мікроконтроллера є окремим агентом, який виконує свою окрему функцію і не є перешкодою для роботи інших агентів.

- Для інтелектуалізації системи було використано IoT-платформу ThingSpeak, яка оптимізувала процес керування сиситемою, та полегшила процес обробки даних. Також був створений мобільний додаток для простішої взаємодії системи та користувача.Система повністю автономна, та не потребує будь-якого втручання людини.

12