Цифровые сенсорные системы

2022/23 учебный год

Инженерный тур

Общая информация

Реализация системы автоматического контроля температуры в помещении для хранения сельскохозяйственной продукции.

Легенда задачи

В летний сезон у фермера Спартана удался урожай яблок. Количество собранных фруктов было таково, что Спартан не сумел их ни продать, ни законсервировать. Было принято решение постараться сохранить яблоки как можно дольше пригодными к употреблению в свежем виде. Для этих целей у фермера нашлось полуподвальное помещение. Помещение не оснащено никакими климатическими установками, что может привести к порче урожая. Спартан предположил, что использование внешних климатических условий в зимний период, может поспособствовать более длительному сроку хранения.

Задумка фермера в том, чтобы запускать принудительно холодный воздух с улицы, при условии, что температура внутри выше, чем на улице. При достижении в помещении необходимой температуры следует прекратить подачу уличного воздуха. В случае, когда внутри помещения температура ниже, чем на улице, нужно просто открыть форточку.

К сожалению, фермер не имеет технических средств и материалов для самостоятельной реализации подобной системы. Ваша задача спроектировать и тических условий.

Требования к команде и компетенциям участников

Количество участников в команде: 4.

Компетенции, которыми должны обладать члены команды: электротехника, программирование микроконтроллеров, кодирование сигнала, алгоритмы, передача данных.

Роли, которые должны быть представлены в команде:

- Сетевик.
- Математик.
- Электротехник.
- Программист микроконтроллеров.

Оборудование и программное обеспечение

Наименование	Описание
Arduino IDE/ESP IDF	ПО для программирования микроконтроллера
Python3	ПО для написания серверной части
Отладочная плата ESP-WROOM-32 DevKit v1	Основной модуль стенда
Микрокомпьютер Raspberry Pi 3	Серверная часть системы
Датчик температуры LM235Z	Аналоговый датчик температуры
Транзистор MOSFET N-CH	Компонент для реализации ШИМ
Резистор 7 кОм	
Вентилятор 12 В	Модуль, отвечающий за охлаждение
Сервопривод	Модуль, имитирующий открытие/закрытие форточки
Светильник с лампой 40 Вт	Модуль, отвечающий за нагревание

Описание задачи

Условие

В летний сезон у фермера Спартана удался урожай яблок. Количество собранных фруктов было таково, что Спартан не сумел их ни продать, ни законсервировать. Было принято решение постараться сохранить яблоки как можно дольше пригодными к употреблению в свежем виде. Для этих целей у фермера нашлось полуподвальное помещение. Помещение не оснащено никакими климатическими установками, что может привести к порче урожая. Спартан предположил, что использование внешних климатических условий в зимний период, может поспособствовать более длительному сроку хранения.

Задумка фермера в том, чтобы запускать принудительно холодный воздух с улицы, при условии, что температура внутри выше, чем на улице. При достижении в помещении необходимой температуры следует прекратить подачу уличного воздуха. В случае, когда внутри помещения температура ниже, чем на улице, нужно просто открыть форточку.

К сожалению, фермер не имеет технических средств и материалов для самостоятельной реализации подобной системы. Ваша задача спроектировать и разработать распределенную систему управления климатом в помещении с использованием внешних климатических условий.

Список оборудования, комплектующих и программного обеспечения

- Отладочная плата ESP-WROOM-32 DevKit v1 — 2 шт.
- Микрокомпьютер Raspberry Pi 3 — 1 шт.
- Персональный компьютер — 2 шт.
- Датчик температуры LM235Z — 2 шт.
- Транзистор MOSFET N-CH — 1шт.
- Резистор 7 кОм — 1 шт.
- Вентилятор 12В — 1шт.
- Сервопривод — 1шт.

- Светильник с лампой 40Вт.
- Arduino IDE.
- ESP IDF.
- Python3.

Техническое задание

Необходимо спроектировать и разработать распределенную сенсорную систему, состоящую из следующих модулей:

- внешний сенсорный модуль;
- внутренний сенсорный и исполнительный модуль;
- центральный модуль, выполняющий роль сервера.

1. Внешний сенсорный модуль должен включать в себя отладочную плату на основе ESP32 и температурный датчик.
2. Внутренний модуль должен включать в себя отладочную плату на основе ESP32, температурный датчик, сервопривод и вентилятор.
3. Центральный модуль должен быть основан на микрокомпьютере Raspberry Pi, агрегировать в себе информацию, получаемую от сенсорных модулей и формировать управляющие воздействия для исполнительных устройств.

Все модули находятся в одной беспроводной локальной сети.

Для осуществления передачи данных между модулями системы необходимо разработать собственный протокол, работающий поверх протокола TCP.

Рекомендации к выполнению

Рекомендуется предусмотреть дифференцированную скорость работы вентилятора в зависимости от величины разницы внешней и внутренней температур.

Рекомендуется сконфигурировать точку доступа на микрокомпьютере. При разработке допускается использование доступной Wi-Fi сети.

Разработку рекомендуется вести с помощью системы контроля версий git.

Этапы решения

Этап 1 Разработка аппаратной части для сенсорных модулей

Для функционирования сенсорных модулей необходимо разработать аппаратную часть, которая предполагает интеграцию с ESP32, а именно: подключить аналоговый датчик температуры, транзистор для управления вентилятором, вентилятор, сервопривод. Необходимо осуществить все соединения на макетной плате.

Результатом будет являться электрическая схема.

Для выявления проблем и отладки необходимо реализоваться действия из второго этапа.

Данный этап необходим для функционирования всей системы. В случае неудачной реализации участники могут смоделировать работу некоторых аппаратных частей.

Этап 2 Разработка программной части для сенсорных модулей

Для полного функционирования сенсорных модулей необходимо разработать программную часть для отладочных план ESP32. Программная часть обрабатывает и формирует сигналы для аппаратной части. Программный код может быть написать как в Arduino IDE так и с помощью любого редактора и «залит» с помощью Espressif.

Результатом будет функционирующий автономно сенсорный модуль.

При разработке могут возникнуть проблемы с калибровкой аналогового датчика. Необходимо изучить документацию на датчик, для задания верный параметров. Также стоит учитывать разрядность АЦП, находящегося на отладочной плате ESP32.

Данный этап необходим для функционирования всей системы. В случае неудачной реализации участники не смогут продемонстрировать полную работу.

Этап 3 Разработка программной части серверного модуля, протокола обмена

Для осуществления сбора данных от сенсорных модулей необходимо разработать программную часть, который будет работать на Raspberry Pi. Это дает полную работу системы. Один из способов: реализация TCP сервера на языке программирование Python. Также необходимо разработать протокол, который работает поверх TCP, для корректного обмена данными.

Результатом станет возможность обмена данными между всеми модулями.

При разработке могут возникнуть проблемы с протоколом, который придется корректировать в зависимости от передаваемых данных.

Данный этап необходим для функционирования всей системы. В случае неудачной реализации участники не смогут продемонстрировать полную работу.

Система оценивания

Для оценки работоспособности устройства выполняется тестирование путем имитации различных условий окружающей среды.

Основными критериями оценки результатов являются правильность определения температуры и смена состояний системы, для ее поддержания.

При прочих равных результатах оценивается:

1. Наличие пользовательского интерфейса на серверном модуле.
2. Скорость реагирования на внешние изменения.
3. Наличие возможности подключения извне к серверному модулю.

№	Критерий	Балл	Комментарий
1	Работа аналогового датчика	10	5 баллов за каждый
2	Пропорциональная работа вентилятора	15	5 баллов, если не пропорциональная
3	Работа сервопривода	5	
4	Данные с сенсорных/исполнительных модулей корректно передаются на центральный модуль	10	
5	Передача на сенсорные модули управляющих воздействий	10	

№	Критерий	Балл	Комментарий
6	Работа системы в целом	45	По 15 баллов за каждое состояние
7	Проект велся в системе контроля версий	5	
8	Реализована точка доступа на микрокомпьютере	5	
Максимальный балл		100	

Решения сдаются путем демонстрирования работоспособности системы экспертам.

Команда-победитель — команда, показавшая наилучший результат по итогам экспертной оценки.

Итоговый индивидуальные балл участника равен:

$$\text{Физика} \times 15\% + \text{Информатика} \times 15\% + \text{Командный балл} \times 70\%.$$

Первые 25% рейтинга участников, построенного согласно итоговым индивидуальным баллам независимо от класса, становятся призерами и победителями, 8% — победителями.

Решение задачи

Дифференцированное изменение вращения происходит по средствам работы с ШИМ. Ссылки на статьи по работе:

- https://habr.com/ru/company/first/blog/664922/.
- https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/mcpwm.html.

Снятие данных с АЦП для датчика температуры: https://docs.espressif.com/projects/esp-idf/en/v4.4/esp32/api-reference/peripherals/adc.html.

Код ESP32 (sensor_client.ino)

```
//    Wi-Fi    //
const char* ssid     = "SPARTAN";
const char* password = "12345678";
const char* host     = "192.168.50.10";
const uint16_t port  = 65434;

#include <WiFi.h>                          // библиотека Wi-Fi
WiFiClient client;
//    Wi-Fi    //

//    setup    //
void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```

```
  Serial.println();
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  client.connect(host, port);
}
//    setup    //

//    loop    //
void loop() {
  float t = read_temp(36);
  Serial.println(t);
  // если есть подключение к серверу
  if (client) {
    // отправляем данные
    client.print(String(t) + ";");
  } else {
    // иначе
    // пробуем подключиться
    if (client.connect(host, port))
      Serial.println("connected");
    else {
      Serial.println("error connect");
      delay(5000);
    }
  }
  delay(200);
}
//    loop    //

//    fun    //
// функция чтения температуры
float read_temp(uint8_t pin) {
  static float t = 20;
  t += ((float)analogReadMilliVolts(pin) / 10.0 - 271 - t) * 0.1;
  return t;
}
//    fun    //
```

Код ESP32 (servo_client.ino)

```
//    Servo    //
#include <ESP32Servo.h>
Servo servo;
#define CLOSE 135
#define OPEN 45
//    Servo    //

//    Wi-Fi    //
const char* ssid     = "SPARTAN";
const char* password = "12345678";
const char* host     = "192.168.50.10";
const uint16_t port  = 65433;

#include <WiFi.h>
WiFiClient client;
//    Wi-Fi    //

//    variables    //
uint8_t pwm;
```

```
bool win;
//    variables    //

//      setup      //
void setup() {
  Serial.begin(115200);

  // инициализация ШИМ
  ledcSetup(4, 500, 6);
  ledcAttachPin(26, 4);

  // инициализация сервопривода
  ESP32PWM::allocateTimer(0);
  servo.setPeriodHertz(50);
  servo.attach(32, 544, 2400);

  // подключение к сети
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
//      setup      //

//    loop    //
void loop() {
  float t = read_temp(34);
  Serial.println(t);
  // если есть подключение к серверу
  if (client) {
    // отправляем данные
    client.print(String(t) + ";");
    // если в буффере есть данные
    if (client.available()) {
      // читаем 1 байт
      uint8_t data = client.read();
      // извлекаем информацию
      win = (data >> 6) & 1;
      // устанавливаем положение сервопривода
      servo.write(win ? CLOSE : OPEN);
      pwm = data & 0x3F;
      // задаём мощность питания вентилятора
      fan_set(pwm);
      // для отладки
      //Serial.println(data, BIN);
      //Serial.println(char(data));
    }
    //Serial.printf("%2.2f\t%d\t%d\n\r", read_temp(34), win, (pwm * 100) / 63);
  } else {
    // иначе
    // пробуем подключиться
    if (client.connect(host, port))
      Serial.println("connected");
    else {
      Serial.println("error connect");
      delay(5000);
```

```
    }
  }
  delay(200);
}
//    loop    //

//    fun    //
// функция чтения температуры
float read_temp(uint8_t pin) {
  static float t = 20;
  t += ((float)analogReadMilliVolts(pin) / 10.0 - 278 - t) * 0.05;
  return t;
}

// функция установки скорости вращения вентилятора
void fan_set(uint8_t val) {
  ledcWrite(4, val);
}
//    fun    //
```

Серверная часть (dashboard.py)

```
from dash import *
import plotly.express as px
import pandas as pd
import plotly.graph_objects as go

def get_num_lines(fname):
    with open(fname) as f:
        c = 0
        for i, _ in enumerate(f):
            c += 1
            pass
        return c + 1

def get_data():
    filename = 'workers/data.csv'

    num_lines = get_num_lines(filename)
    n = 3000
    return pd.read_csv('workers/data.csv', delimiter=';',
                       names=["time", "sensor", "data"],
                       skiprows=range(0, num_lines - n))

app = Dash(__name__)

app.layout = html.Div([
    html.H1(children='Национальная технологическая олимпиада', style={'textAlign':
    ↪  'center', 'font-family': 'Gilroy-Medium'}),
    dcc.Dropdown(['sensor', 'servo', 'cooler'], 'servo', id='dropdown-selection'),
    dcc.Graph(id='temperature-content'),
    dcc.Interval(
            id='interval-component',
            interval=1*500, # in milliseconds
            n_intervals=0
        )
])

@callback(
```

```
    Output('temperature-content', 'figure'),
    Input('interval-component', 'n_intervals')
)
def update_temperature(n):
    df = get_data()
    dff = df.loc[df['sensor'] != 'cooler']

    fig = px.line(dff, x='time', y='data', color='sensor')
    return fig

if __name__ == '__main__':
    app.run_server(debug=True)
```

Репозиторий проекта

https://disk.yandex.ru/d/Aq8fEGrBmRHvRQ.

Фото устройства

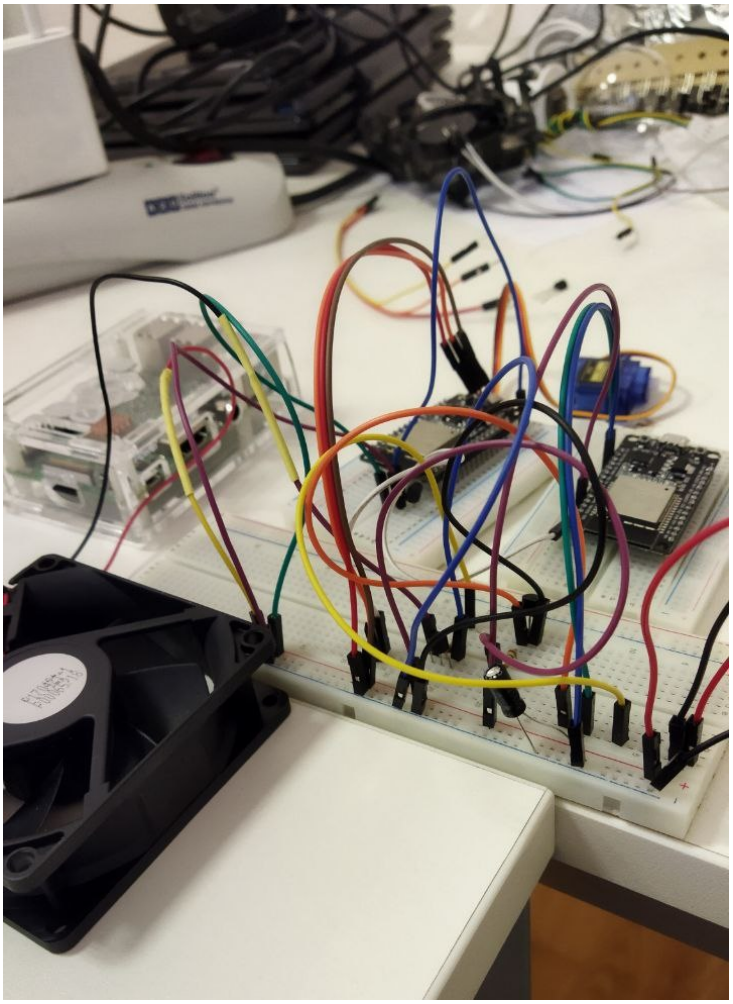

Рис. VI.2.1. Собранное устройство

Снимок экрана из Arduino IDE

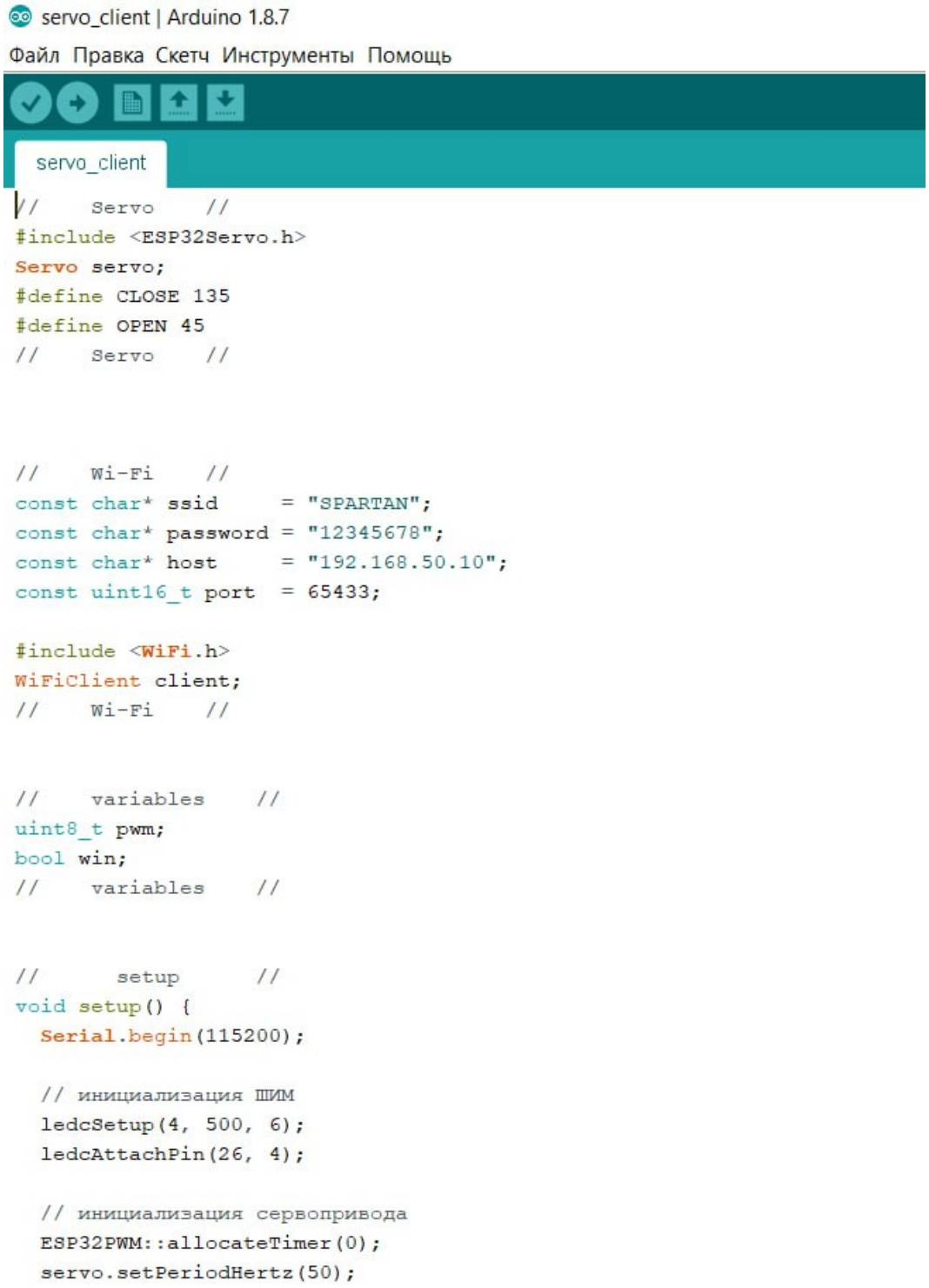

Рис. VI.2.2. Снимок экрана из Arduino IDE

Материалы для подготовки

1. Умняшкин С.В. Основы теории цифровой обработки сигналов: учебное пособие. Москва: ТЕХНОСФЕРА, 2019. — 550 с.
2. Сайт Espressif, информационный лист об API-функции АЦП для СнК ESP32

URL: https://docs.espressif.com/projects/esp-idf/en/v4.4/esp32/api-reference/peripherals/adc.html (дата обращения 20.03.2023).

3. Сайт Chipdip, информационный лист на прибор LM235Z URL: https://static.chipdip.ru/lib/908/DOC018908677.pdf (дата обращения 20.03.2023).
4. Сайт Espressif с документацией на TCP/IP: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/lwip.html (дата обращения 21.03.2023).
5. Сайт Espressif, информационный лист об API-функции ШИМ для управления LED для СнК ESP32 URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/ledc.html (дата обращения 20.03.2023).
6. Сайт Espressif, информационный лист об API-функции ШИМ для управления электродвигателем для СнК ESP32 URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/mcpwm.html (дата обращения 20.03.2023).
7. Хоровиц П., Хилл У. Искусство схемотехники: Пер. с англ. — Изд. 2-е. — М.: Издательство БИНОМ. — 2016. — 704 с.
8. Сайт HABR «ШИМ в ESP32» URL: https://habr.com/ru/company/first/blog/664922/ (дата обращения 20.03.2023).
9. Сайт Электросам URL: https://electrosam.ru/glavnaja/jelektrotehnika/shirotno-impulsnaia-moduliatsiia/ (дата обращения 20.03.2023).
10. Сайт WikiAmperka URL: http://wiki.amperka.ru/articles:servo (дата обращения 21.03.2023).
11. Информационный лист для сервопривода TowerPro MG995 URL: https://www.electronicoscaldas.com/datasheet/MG995_Tower-Pro.pdf (дата обращения 21.03.2023).
12. Миндеева А.А. Микросхемотехника: учеб. пособие. — Изд. 2-е. — М.: МИЭТ, 2016. —188 с.