ЛАБОРАТОРНАЯ РАБОТА №8

Программирование микроконтроллеров

Необходимость выполнения работы: На сегодняшний день многие простые передатчики и приемники строятся на базе микроконтроллеров. Например, ESP32 – двухъядерный микроконтроллер с тактовой частотой до 240 МГц, с встроенными Wi-Fi и Bluetooth модулями. Интернет вещей IoT непосредственно связан с программированием микроконтроллеров.

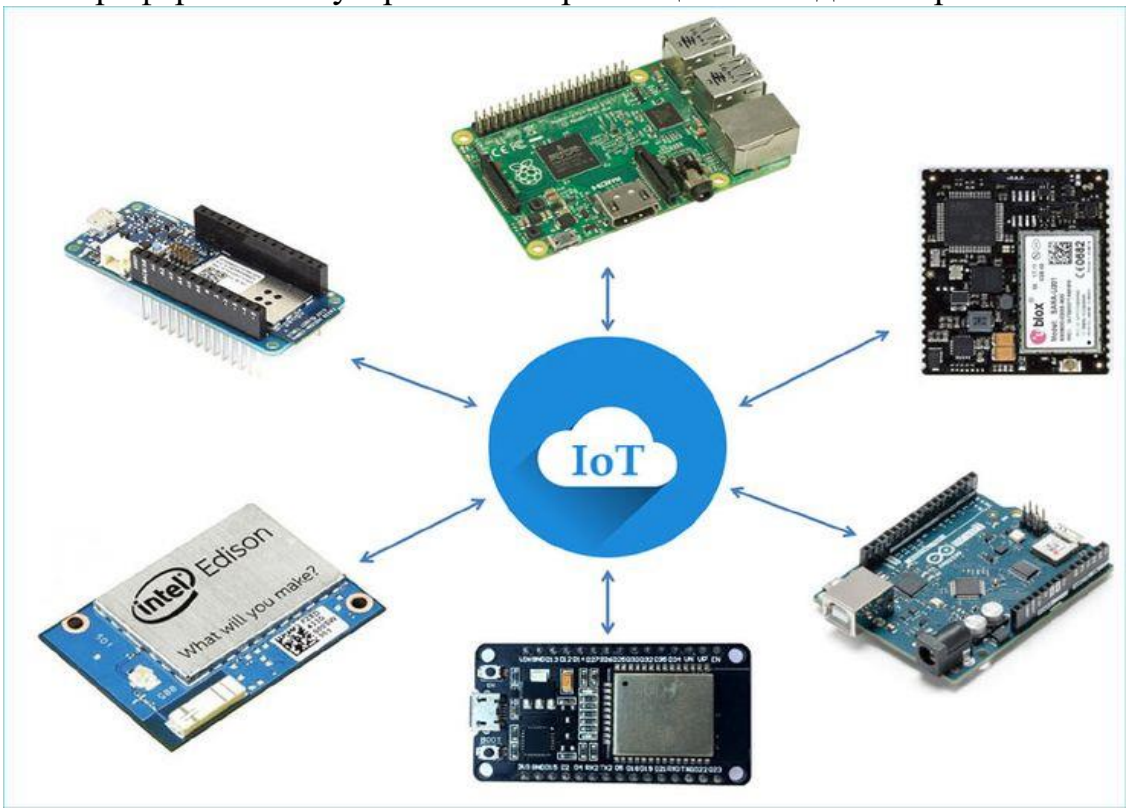
1 Цель работы: Изучение особенностей микроконтроллеров. Получение навыков программирования микроконтроллеров.

2. Содержание отсчета:

1. Титульный лист
2. Цель работы
3. Ответы на вопросы допуска
4. Листинг кода с комментариями

Что такое микроконтроллер?

Микроконтроллер (англ. Micro Controller Unit, MCU) – микросхема, предназначенная для управления электронными устройствами. Данная микросхема работает в соответствии с заложенной в нее программой, которую создает программист. В отличие от компьютера в микроконтроллере ядро, память и управление периферийными устройствами размещены на одном кристалле.



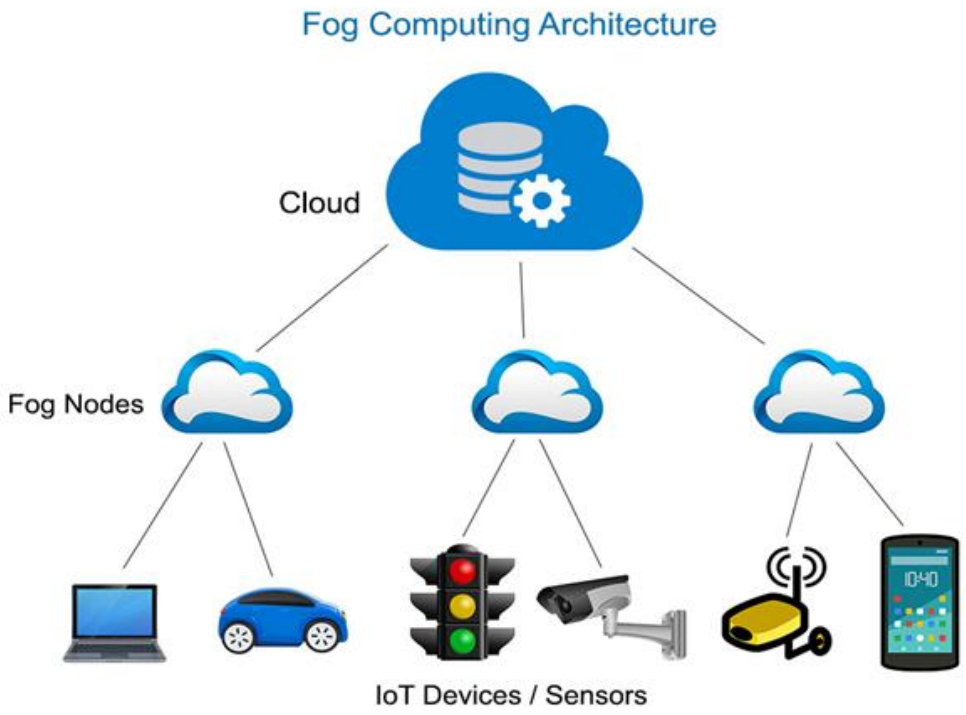
В рукописной форме ответить на вопросы допуска к работе:

1. Какие языки программирования наиболее распространены для программирования микроконтроллеров и в чем их ключевые различия?
2. Почему язык С считается основным для программирования микроконтроллеров? Какие особенности С делают его идеальным для встраиваемых систем?
3. Какие основные элементы входят в микроконтроллер и их назначение.
4. Как прошить микроконтроллер?
5. Какие бывают семейства микроконтроллеров?
6. Чем микроконтроллер отличается от компьютера?
7. Примеры использования микроконтроллеров
8. Что означает «программирование встраиваемых систем»?
9. Как объем оперативной памяти у микроконтроллеров?

Программирование микроконтроллеров в сети Интернета вещей

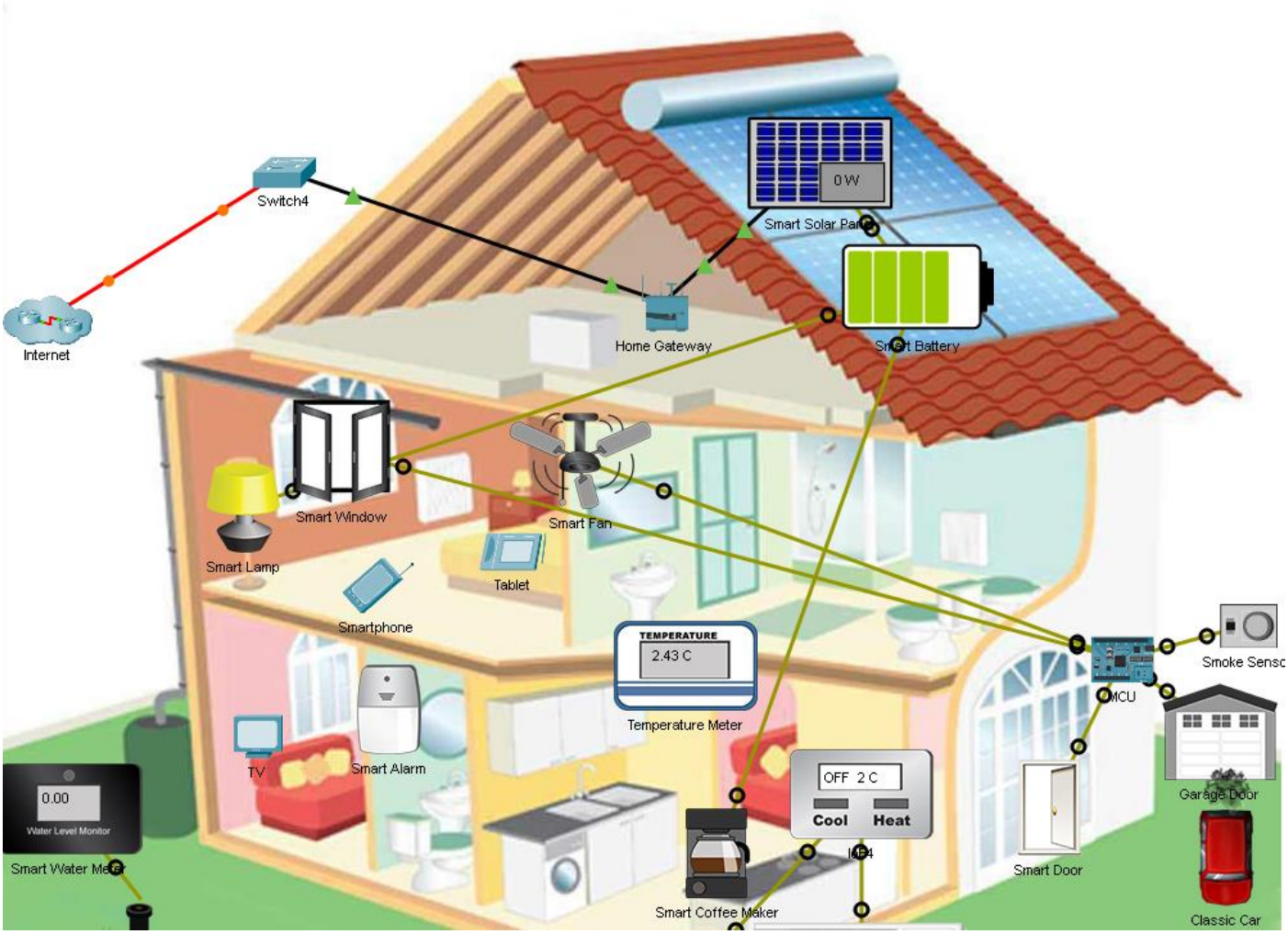
Задание 1. Реализуйте «Туманные вычисления» в сети IoT. У нас имеется «умный дом». В MCU во вкладке Programming напишите скрипт ни языке Python для открывания и закрывания гаражных ворот, входной двери и окна, а также для включения и выключения вентилятора в зависимости от уровня угарного газа в доме.

Туманные вычисления» в сети IoT



Fog computing (туманные вычисления) – это децентрализованная вычислительная инфраструктура, в которой данные, вычисления, хранилища и приложения расположены где-то между источником данных и облаком.

Метафора тумана происходит от метеорологического термина, обозначающего облако, расположенное близко к земле, точно так же, как туман концентрируется на границе сети.



Владелец держит в домашнем гараже машину. Машина выделяет угарный газ, который через дверь частично попадает в помещение жилого дома.

MCU, добавленный к умному дому, используется для контроля уровня дыма, считываемого датчиком дыма (Smoke Sensor), и принятия решения о необходимости проветривания дома. Если уровень угарного газа поднимается выше 10,3 единиц, MCU программируется на автоматическое открытие окна, входной двери, гаражных ворот и запуск вентилятора на высокой скорости. Это действие отменяется только тогда, когда уровень угарного газа опускается ниже 1 единицы.

Пример кода на Java Script для гаражных ворот:

```
var smk_sensor = A0;
var garDoor = 5;

function setup() {
```

```
     pinMode(garDoor, OUTPUT);
     }

function loop() {
     // считывание с датчика
     var newValue = analogRead(smk_sensor);
     newValue = newValue/10;


     if (newValue > 10) {
          customWrite(garDoor, 1);

     } else {
          if (newValue < 1) {
            customWrite(garDoor, 0);
          }
     }
     Serial.println("Уровень угарного газа: " + newValue);
}
```

Пример части кода для датчика дыма на Python
файл main.py:

```
from gpio import *
from time import *
from environment import *
from physical import *
from pyjs import *
import math

ENVIRONMENT_NAME = "Smoke"
MIN = 0
MAX = 100


def main():
    while True:
        loop()

def loop():
    value = float(Environment.get(ENVIRONMENT_NAME))
    #print(value)
    if value < MIN:
```

```
            value = MIN
        elif value > MAX:
            value = MAX

        setDeviceProperty(getName(), "level", value)
        value = math.floor(js_map(value, MIN, MAX, 0, 255))
        analogWrite(A0, value)
        sleep(1)

    if __name__ == "__main__":
        main()
```

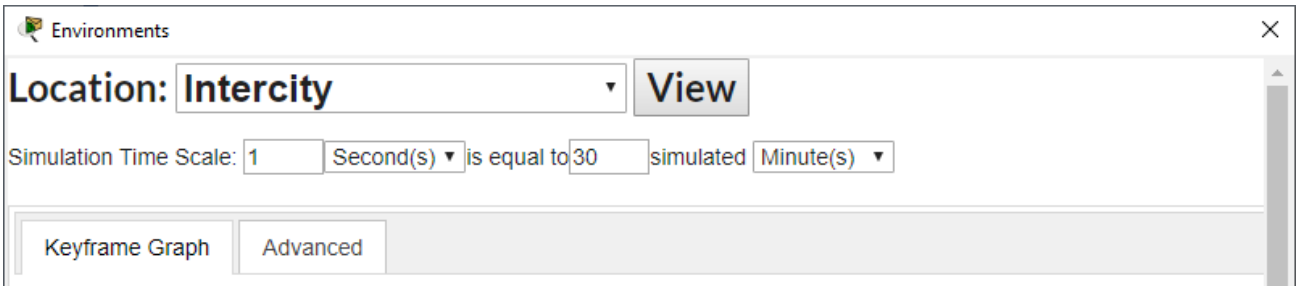
файл pyjs.py:

```
    class JsObject(dict):
        def __init__(self, d):
            for k in d.keys():
                setattr(self, k, d[k])

    def js_map(x, inMin, inMax, outMin, outMax):
        return (x - inMin) * (outMax - outMin) / (inMax - inMin) +
outMin
```

Не забудьте запустить выполнение кода кнопкой Run.

В правом верхнем углу щелкните на Environment, вверху где Location -> Edit и вместо Sumilation Time 1 минута выберите 1 секунда



Закройте окно, закройте программу Packet Tracer сохранив файл. Откройте файл заново.

Нажмите на планшет.

Перейдите в раздел Рабочий стол > Веб-браузер.

В адресной строке введите 192.168.25.1. Это IP-адрес домашнего шлюза.

Используйте admin/admin в качестве имени пользователя и пароля для входа в домашний шлюз.

Щелкните по детектору дыма; оставьте это окно видимым, чтобы вы могли контролировать уровень задымленности.

Запустите двигатель, удерживая клавишу Alt и нажав на значок автомобиля.

Что происходит с воздухом в доме, когда машина стоит в гараже? Датчик дыма показывает повышенный уровень угарных газов. Когда уровень превышает 10,3 единицы, MCU реагирует на него и открывает гаражные ворота, входную дверь и окно. MCU также включает потолочный вентилятор на максимальной скорости.

Продолжая следить за уровнями, остановите двигатель автомобиля, удерживая клавишу Alt и щелкнув по автомобилю.

Подумайте, что происходит с дверями, окном и вентилятором? Теперь, когда уровни ниже 1 единицы, MCU решает, что можно безопасно закрыть гаражные ворота, входную дверь и окно. MCU также останавливает вентилятор.

Этот пример показывает, что выбор между облачными вычислениями и туманными зависит от приложения.

В данном примере с умным домом и угарным газом наилучшим вариантом были туманные вычисления. Данные, полученные от датчиков дыма, обрабатывались и использовались для принятия решений относительно качества воздуха в доме. В этом сценарии не было необходимости отправлять данные датчиков в облако для обработки. Обработка в облаке замедлила бы время отклика, что потенциально подвергало бы жизни опасности. Другая возможная проблема связана с интернет-соединением: если бы соединение с Интернетом было потеряно, вся система вышла бы из строя, что поставило бы жизни под угрозу.

Перепишите код с Java Script, Python на язык Си и скомпилируйте в исполняемый машинный код (файл .hex) для прошивки микроконтроллера под плату Arduino Uno с помощью Arduino IDE.

Задание 2. Напишите программный код на языке Python для микроконтроллера MCU (им может быть например ESP8266, ESP32, Atmega16) и одноплатного компьютера (им может быть например Raspberry PI B+ или Orange Pi) для автоматизации для подачи пожарной тревоги при обнаружении пожара удаленным пожарным датчиком.

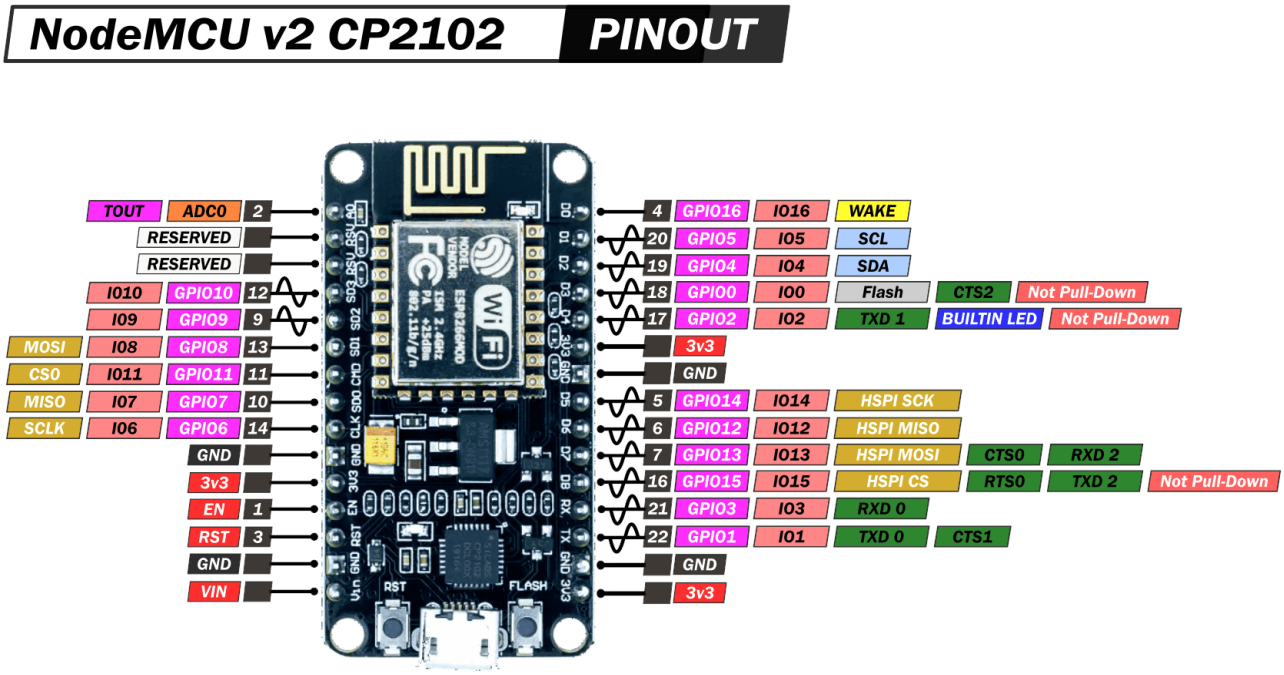


Рисунок 3. – Плата NodeMCU на базе микроконтроллера ESP8266

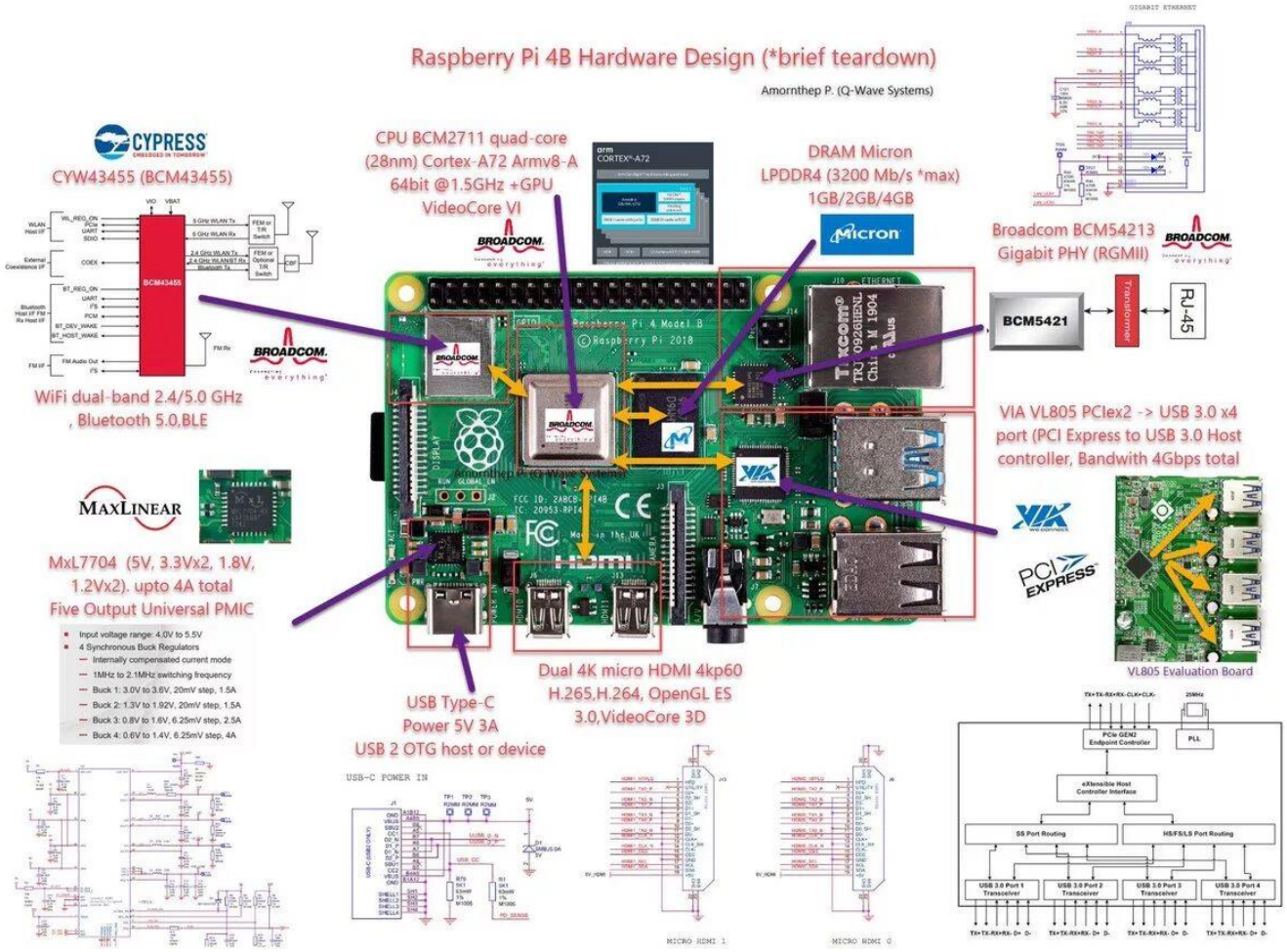


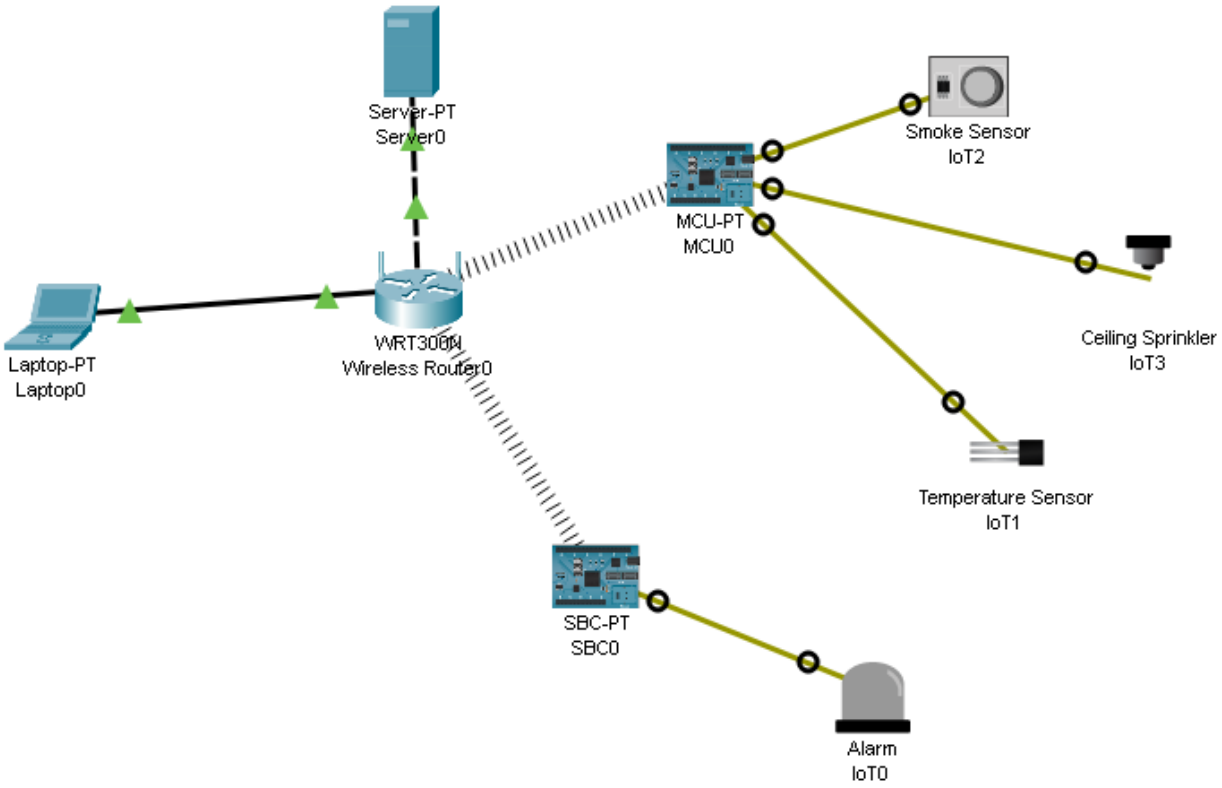
Рисунок 4. – Одноплатный компьютер Raspberry PI

Плата микроконтроллера, подключенная к датчикам дыма и температуры, будет действовать как устройство дистанционного обнаружения пожара. Плата будет надежно подключена по Wi-Fi к маршрутизатору.

SBC bord, подключенный к визуальной сигнализации, будет действовать как система пожарной сигнализации.

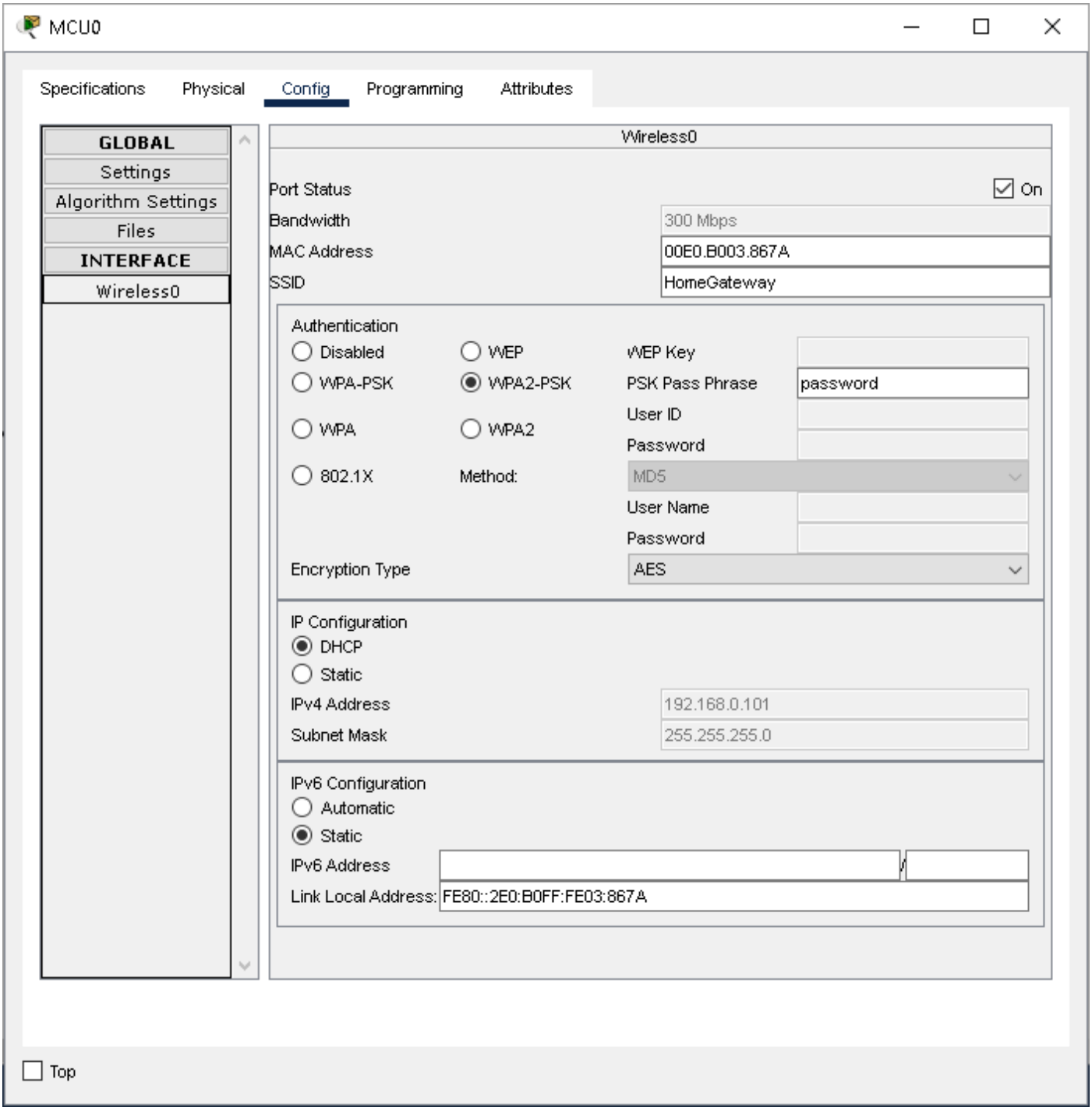
Обе платы Интернета вещей будут зарегистрированы на центральном сервере IoT/IoE, который предоставит возможности автоматизации для подачи пожарной тревоги при обнаружении пожара удаленным пожарным датчиком.

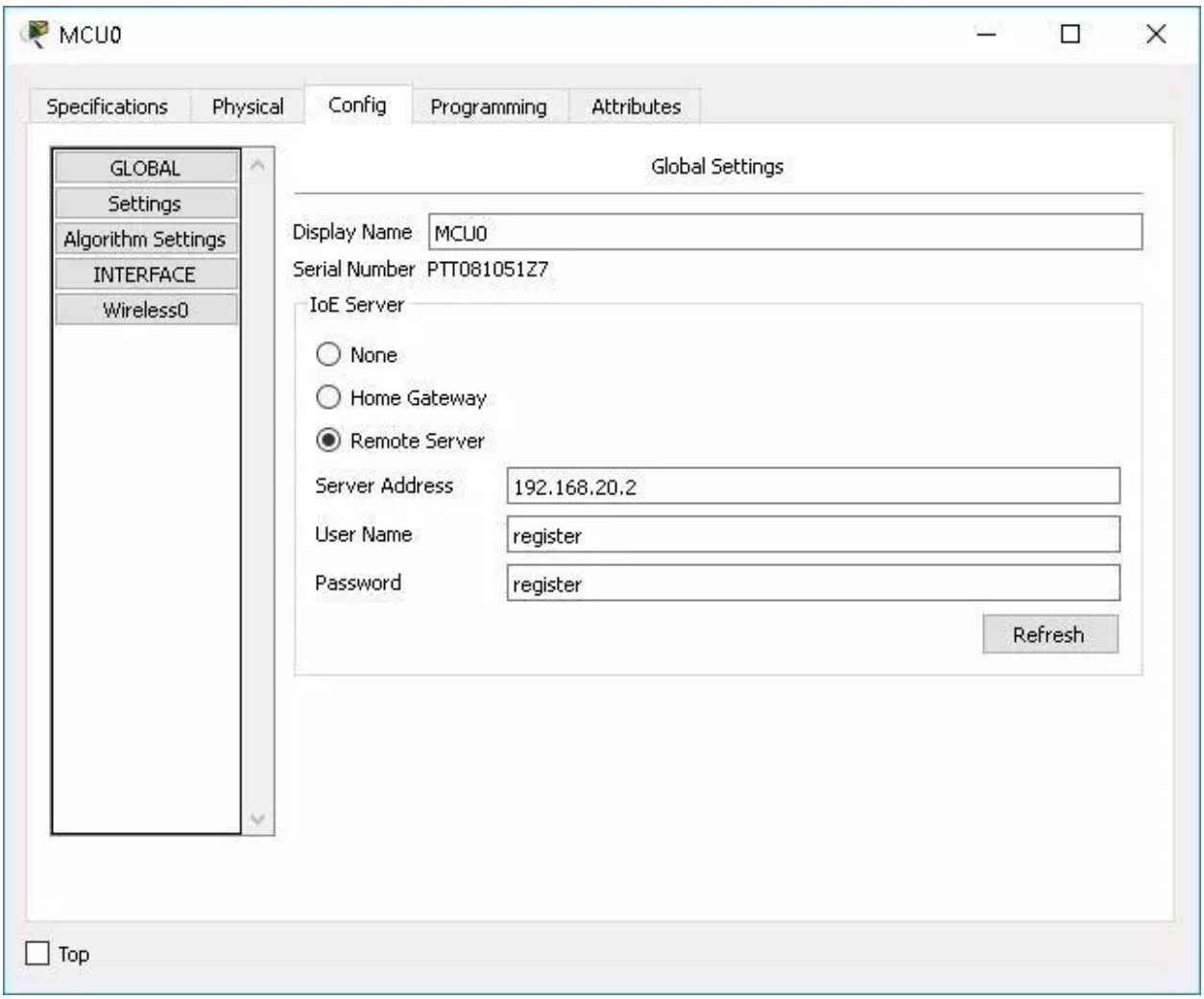
Собираем схему:



Регистрация платы IoT/IoE MCU

Добавьте беспроводной разъем к плате MCU и настройте параметры беспроводной связи для подключения его к удаленному маршрутизатору. На вкладке config настройте адрес удаленного сервера IoT/IoE (192.168.20.2) со следующим пользователем/паролем (user : register, пароль : register)


MCU0
Specifications
Physical
Config
Programming
Attributes
GLOBAL
Settings
Algorithm Settings
Files
INTERFACE
Wireless0
Wireless0
Port Status
On
Bandwidth
300 Mbps
MAC Address
00E0.B003.867A
SSID
HomeGateway
Authentication
Disabled
WEP
WEP Key
WPA-PSK
WPA2-PSK
PSK Pass Phrase
password
User ID
WPA
WPA2
Password
802.1X
Method:
MD5
User Name
Password
Encryption Type
AES
IP Configuration
DHCP
Static
IPv4 Address
192.168.0.101
Subnet Mask
255.255.255.0
IPv6 Configuration
Automatic
Static
IPv6 Address
Link Local Address:
FE80::2E0:B0FF:FE03:867A
Top



Код Javascript для платы MCU (дистанционные датчики пожара)

Функция IoT/IoEClient.setup(api) настраивает API для удаленного мониторинга и управления с IoT/IoE-сервера. API - это объект, описывающий состояния, сообщаемые устройством ввода-вывода, и идентифицирующий одно удаленное управляемое сервером.

IoT/IoEClient.reportStates(состояния) сообщает о состояниях этого устройства серверу ввода-вывода. Аргумент представляет собой строку, представляющую все состояния этого устройства. Аргументом также может быть массив, представляющий все состояния. Количество состояний должно соответствовать длине свойства states в функции IoT/IoEClient.setup().

```
function setup() {
      pinMode(1, OUTPUT);
      IoEClient.setup({
       type: "Fire Detection",
       states: [{
            name: "Fire",
            type: "bool"
```

```
        },
        {
        name: "Temperature",
        type: "number",
        unit: "°C",
        imperialUnit: "°F",
        toImperialConversion: "x*1.8+32",
        toMetricConversion: "(x-32)/1.8",
        decimalDigits: 1
       },
       {
        name: "Smoke Level",
        type: "number",
        unit: "ppm",
        decimalDigits: 1
       }]
      });
}

function loop() {
      var SmokeLevel = analogRead(A0);
      var Temperature = analogRead(A1);
      var FireDetected=false;

      if (SmokeLevel>50 || Temperature>580) {
      digitalWrite(0, HIGH);
      FireDetected=true;
      }
      else {
      digitalWrite(0, LOW);
      FireDetected=false;
      }
    IoEClient.reportStates([FireDetected,Temperature,SmokeLevel]);
      delay(500);
}
```

Код Javascript для платы SBC (визуальная пожарная сигнализация)

```
var state = 0;

function setup() {

      IoEClient.setup({
            type: "FireAlarm",
            states: [{
                  name: "On",
                  type: "bool",
                  controllable: true
            }]
      });

      IoEClient.onInputReceive = function(input) {
            processData(input, true);
      };
```

```
	attachInterrupt(0, function() {
		processData(customRead(0), false);
	});

	setState(state);
	}

function processData(data, bIsRemote) {
	if ( data.length <= 0  )
		return;
	setState(parseInt(data));
}

function setState(newState) {
	state = newState;

	if ( state === 0 )
		digitalWrite(0, LOW);
	else
		digitalWrite(0, HIGH);

	customWrite(0, state);
	IoEClient.reportStates(state);
	setDeviceProperty(getName(), "state", state);
}
```

Задание 3. Создайте умное окно в рамках Интернета вещей. Окно должно отслеживать уровень дождя в районе и при обнаружении дождя закрываться само. Ниже приведено несколько советов:

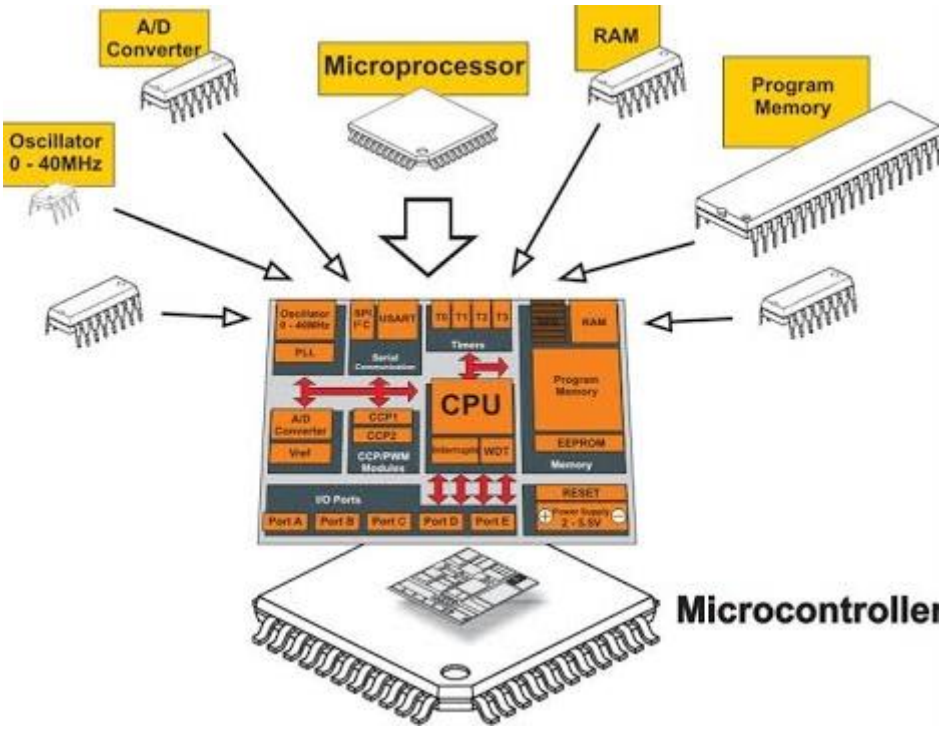
- Используйте обычное окно из стандартного набора компонентов эмулятора CPT.
- Используя вкладку "Программирование" в настройках окна, отредактируйте код для обнаружения дождя и принятия мер в случае его возникновения.
- Окно должно запомнить свое последнее состояние (закрыто или открыто). Когда дождь прекратится, окно должно вернуться в открытое состояние.
- Используйте среду эмулятора CPT - вызов Javascript API Environment.get(“Wind Speed”) для считывания значения дождя из моделируемой среды эмулятора CPT.

Приложение

Microprocessor Microcontroller

- микроконтроллер — однокристальное устройство
- ресурсы программы << ресурсы микропроцессора
- ресурсы программы ≈ ресурсы микроконтроллера

	MCU	CPU
ОЗУ	встроенное	внешнее
Объём ОЗУ	< 1 МБ	>> 1 МБ
Постоянная память	встроенная	внешняя
Объём памяти	< 1 МБ	>> 1 МБ
Периферийные устройства	в основном встроенные	в основном внешние



Все элементы на одном кристалле (подложке) в одном корпусе

Контрольные вопросы:

1. Какие преимущества у ESP8266 по сравнению с другими микроконтроллерами?
2. Перечислите основные протоколы связи в микроконтроллерах и их применение
3. В каких случаях используют ассемблер для программирования МК, несмотря на его сложность?
4. Чем архитектура микроконтроллера принципиально отличается от архитектуры микропроцессора общего назначения?
5. Какие типы памяти используются в микроконтроллерах и какова их назначение?
6. Что означает термин "система на кристалле" (SoC) применительно к микроконтроллерам?
7. Опишите процесс разработки программы для микроконтроллера.
8. Какие факторы влияют на выбор конкретного микроконтроллера для определенного проекта?
9. Какие компании разрабатывают микроконтроллеры?
10. Что такое регистры микроконтроллера и как осуществляется работа с ними на уровне программирования?