**Преподавание основ алгоритмизации через программирование микроконтроллеров**

Образовательная система стремится к междисциплинарному подходу, где знания и навыки из разных предметов синтезируются для достижения более глубокого понимания и развитие компетенций XXI века. Объединение предметов "Основы алгоритмизации" и "Программирование микроконтроллеров" является логичным шагом в этом направлении, позволяя ученикам не только изучить основы программирования, но и применить их на практике, создавая реальные проекты с использованием микроконтроллеров.

Современные студенты, выросшие в окружении постоянного потока информации, часто сталкиваются с проблемой клипового мышления. Это проявляется в трудности сосредоточиться на одном задании длительное время и глубоко погрузиться в процесс обучения.

**Причин этого явления несколько:**

* **Постоянное воздействие коротких фрагментов информации:** Социальные сети, видеоплатформы, новости - все это формирует привычку воспринимать информацию в виде коротких фрагментов, без глубокого анализа и размышления
* **Отсутствие навыков глубокого чтения и анализа:** Современные учащиеся часто привыкли к "пробежке по тексту", не уделяя внимания деталям и глубокому пониманию материала.
* **Высокая скорость информационного потока:** Постоянное воздействие новой информации приводит к тому, что студентам трудно сохранять сосредоточение на одной задаче длительное время.
* **Недостаток опыта сосредоточенной работы:** В реальности многие студенты не имеют достаточного опыта сосредоточенной работы над одним заданием длительное время, что делает им трудным погрузиться в процесс обучения.

**Клиповое мышление оказывает негативное влияние на успеваемость студентов, снижая их способность к глубокому анализу информации, решению сложных задач и усвоению знаний.**

Изучение дисциплины программирование микроконтроллеров требует построения последовательных шагов (алгоритмов) для решения задачи, что позволяет ученикам отказаться от поверхностного восприятия информации и научиться анализировать процессы в целом. А мгновенно видеть результаты своих алгоритмов и решений очень важно при неумении сосредоточится на проблеме.

Это все помогает студентам уменьшить негативное действие клипового мышления:

1. Укрепить взаимодействие между теорией и практикой: наблюдая за тема, как их алгоритмы влияют на физические устройства, они лучше понимают абстрактные концепции
2. **Развить интуитивное понимание программирования:** визуализация результатов кода делает процесс обучения более интерактивным и интересным, позволяя студентам быстро получать обратную связь и видеть как их решения работают на практике.
3. **Повысить мотивацию и интерес к обучению:** мгновенная обратная связь делает программирование более увлекательным и позволяет студентам видеть свои успехи и результаты своих трудов.
4. **Развить навыки отладки и исправления ошибок:** видеть результат своих действий в реальном времени позволяет быстро определить ошибки в коде и найти правильное решение.
5. **Повысить уровень понимания и освоения алгоритмов:** программирование микроконтроллеров требует четкого понимания алгоритмов и последовательности действий. Мгновенная обратная связь позволяет студентам видеть как работает их алгоритм и как изменения в коде влияют на результат.

Основные этапы обучения:

**Выбор платформы**

Наиболее популярные платформы для работы с микроконтроллерами включают Arduino, Raspberry Pi и STM32. Arduino — отличный старт для новичков благодаря простоте использования и обширной документации.

**Установка среды разработки**

Для Arduino необходимо установить Arduino IDE, которая предоставляет удобный интерфейс для написания и загрузки кода на плату.

**Знакомство с микроконтроллерами**

 Объяснить основные принципы работы микроконтроллеров, их архитектуру, входы-выходы.

**Основы программирования**

Студенты начинают с простых задач, таких как мигание светодиодом. Это помогает понять базовые команды и структуру программы:

C++

void setup() {

 pinMode(LED\_BUILTIN, OUTPUT);

}

void loop() {

 digitalWrite(LED\_BUILTIN, HIGH); *// Включить светодиод*

 delay(1000); *// Подождать 1 секунду*

 digitalWrite(LED\_BUILTIN, LOW); *// Выключить светодиод*

 delay(1000); *// Подождать 1 секунду*

}

Когда ученики усвоят основы, вводить новые концепции: циклы, условия, функции, массивы, объектно-ориентированное программирование

После освоения основ студенты могут перейти к более сложным проектам, таким как:

* Автоматизированная система полива растений: использование датчиков влажности и реле для управления насосом.
* Робот на колесах: создание простого робота с использованием датчиков расстояния и управления движением.

Рабочая программа может состоять из следующих модулей

**Модуль 1 Основы алгоритмизации (8 часов)**

1. Понятие алгоритма, его свойства. Сущность алгоритмизации вычислительного процесса. Способы описания алгоритмов
2. Структурный подход к разработке алгоритмов. Базовые алгоритмические структуры. Программные средства для графического отображения алгоритмов
3. Основные базовые типы данных и их характеристика. Структурированные типы данных и их характеристика. Понятие массива. Основные алгоритмы обработки массивов
4. Понятие подалгоритма. Модульный подход к разработке алгоритмов. Правила выделения и оформления подалгоритма. Выполнение подалгоритма.Функции и процедуры

**Модуль 2 Знакомство с микроконтроллерами (8 часов)**

1. Основные сведения о работе микроконтроллеров
2. Микроконтроллеры семейства AVR

**Модуль 3 Программирование на языке Си (46 часов)**

1. Изучение основ синтаксиса языка программирования в Arduino IDE.
2. Изучение типов данных
3. Условный оператор. Оператор выбора
4. Циклы с постусловием, с предусловием. С параметром. Вложенные циклы
5. Функции

**Модуль 4. IoT Работа с esp32 ( 70 часов)**

1. Микроконтроллер esp 32, обзор прошивок
2. Сохранение данных в облако
3. Взаимодействие с веб-серверами
4. Отправка уведомлений по электронной почте
5. Отправка данных в sms и push уведомлений
6. Создание телеграмм- бота
7. Управление устройствами через облако

**Пример работы студентов**

*Проект управление теплицей с помощью Bluetooth*

Этапы работы

1. Подключение к плате аналогово датчика температуры

Разработанная схема представлена на рисунке 1.



1. После подключения датчика и тестирования работы датчика, в схему был добавлен еще один датчик, необходимы для работы умной теплицы - Soil Moisture Sensor, датчик влажности почвы (Рисунок 2).



Рисунок 2 – Подключение датчика влажности почвы

1. Так же еще одним необходимым датчиком для умной теплицы был фоторезистор (Рисунок 3). В создаваемой умной теплице он будет позволять отслеживать уровень освещённости.



Рисунок 3 – Добавление фоторезистора

1. После проверки работы фоторезистора, и всей созданной системы, исходя из требований теплицы, было необходимо добавить ещё один датчик, который будет отслеживать температуру воздуха и влажность. Для этого был выбран датчик DH11 (Рисунок 4).



Рисунок 4 – Подключение датчика DH11

1. Для наглядного отображения управления освещением был добавлен светодиод (Рисунок 5).



Рисунок 5 – Добавление светодиода

1. Для дистанционного управления был выбран Bluetooth-модуль. Поскольку существует несколько видов Bluetooth-модулей были рассмотрены HC-05 и HC-06, которые являются наиболее используемыми их технические характеристики и выбран более подходящий.

 Подключение обоих модулей к плате Arduino одинаково (Рисунок 6).



Рисунок 6 – Подключение модуля Bluetooth HC-05 к плате

Приложение для управления



Рисунок 7 – Мобильное приложения для управления умной теплицей

**Листинг для проверки подключения датчиков температуры**

//Подключение датчиков к плате и проверка их работы. Тестовая программа.

// подключение библиотеки SoftwareSerial

#include <SoftwareSerial.h>

// тип датчика DHT

#define DHTTYPE DHT11

// контакт подключения аналогового выхода модуля влажности почвы

int  pinSoilMoisture=A0;

// контакт подключения аналогового выхода датчика температуры TMP36

int  pinTMP36=A1;

// контакт подключения аналогового выхода фоторезистора

int pinPhotoresistor=A2;

unsigned long millisupdate=0;

// для получения данных из SoftwareSerial

String inputString0 = "";

// признак конца полученной строки

boolean stringComplete0 = false;

void setup()

  {

  // запуск последовательного порта

  Serial.begin(9600);

  // запуск SoftwareSerial

  HC05Serial.begin(9600);

  // резервирование 50 bytes для the inputString:

  inputString0.reserve(50);

  }

void loop()

  {

  // ожидание конца строки для анализа поступившего запроса:

  serialEvent0();

  if (stringComplete0)

   {

    Serial.println(inputString0);

    parse\_string0(inputString0);

    // очистить :

    inputString0 = "";

    stringComplete0 = false;

   }

    // получение значения с аналогового вывода модуля влажности почвы

    int val0=analogRead(pinSoilMoisture);

    Serial.print("SoilMoisture= "); Serial.println(val0);

    HC05Serial.print("SM=");

    //HC05Serial.print(h);

    HC05Serial.println();

    delay(10);

    // получение значения с аналогового вывода датчика температуры TMP36

    int val1=analogRead(pinTMP36);

    // перевод в мВ

    int  mV= val1\*1000/1024;

    // перевод в градусы цельсия

    int t=(mV-500)/10+75;//t=23;

    Serial.print("TempTMP36= "); Serial.print(t);Serial.println("  C");

    HC05Serial.print("aT=");HC05Serial.print(t);HC05Serial.println();

    delay(10);

    // получение значения с аналогового вывода фоторезистора

    int val2=analogRead(pinPhotoresistor);

    Serial.print("Light= "); Serial.println(val2);

    HC05Serial.print("Ph=");HC05Serial.print(val2);HC05Serial.println();

    delay(10);

    // обновить

    Serial.println();

    }

}

# Листинг кода для умной теплицы

String statusChWriteKey = "
YYOLRB5YKBX6AZ3V";

#include <SoftwareSerial.h>

SoftwareSerial EspSerial(6, 7);

#define HARDWARE\_RESET 8

*// DS18B20*

#include <OneWire.h>

#include <DallasTemperature.h>

#define ONE\_WIRE\_BUS 5

OneWire oneWire(ONE\_WIRE\_BUS);

DallasTemperature DS18B20(&oneWire);

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)*

#define ldrPIN 1

int light = 0;

*// Soil humidity*

#define soilHumPIN 0

int soilHum = 0;

long writeTimingSeconds = 17;

long startWriteTiming = 0;

long elapsedWriteTime = 0;

boolean pump = 0;

boolean lamp = 0;

int spare = 0;

boolean error;

void setup()

{

 Serial.begin(9600);

 pinMode(HARDWARE\_RESET,OUTPUT);

 digitalWrite(HARDWARE\_RESET, HIGH);

 DS18B20.begin();

 dht.begin();

 EspSerial.begin(9600);

 EspHardwareReset();

 startWriteTiming = millis();

}

void loop()

{

 start: *//label*

 error=0;

 elapsedWriteTime = millis()-startWriteTiming;

 if (elapsedWriteTime > (writeTimingSeconds\*1000))

 {

 readSensors();

 writeThingSpeak();

 startWriteTiming = millis();

 }

 if (error==1)

 {

 Serial.println(" <<<< ERROR >>>>");

 delay (2000);

 goto start; }

}

void readSensors(void)

{

 airTemp = dht.readTemperature();

 airHum = dht.readHumidity();

 DS18B20.requestTemperatures();

 soilTemp = DS18B20.getTempCByIndex(0);

 light = map(analogRead(ldrPIN), 1023, 0, 0, 100); *//LDRDark:0 ==> light 100%*

 soilHum = map(analogRead(soilHumPIN), 1023, 0, 0, 100);

}

void writeThingSpeak(void)

{

 startThingSpeakCmd();

 *// preparacao da string GET*

 String getStr = "GET /update?api\_key=";

 getStr += statusChWriteKey;

 getStr +="&field1=";

 getStr += String(pump);

 getStr +="&field2=";

 getStr += String(lamp);

 getStr +="&field3=";

 getStr += String(airTemp);

 getStr +="&field4=";

 getStr += String(airHum);

 getStr +="&field5=";

 getStr += String(soilTemp);

 getStr +="&field6=";

 getStr += String(soilHum);

 getStr +="&field7=";

 getStr += String(light);

 getStr +="&field8=";

 getStr += String(spare);

 getStr += "\r\n\r\n";

 sendThingSpeakGetCmd(getStr);

}

void EspHardwareReset(void)

{

 Serial.println("Reseting.......");

 digitalWrite(HARDWARE\_RESET, LOW);

 delay(500);

 digitalWrite(HARDWARE\_RESET, HIGH);

 delay(8000);

 Serial.println("RESET");

}

void startThingSpeakCmd(void)

{

 EspSerial.flush();

 String cmd = "AT+CIPSTART=\"TCP\",\"";

 cmd += "184.106.153.149";

 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 ==> lenght cmd: ");

 Serial.println(cmd);

 if(EspSerial.find((char \*)">"))

 {

 EspSerial.print(getStr);

 Serial.print("enviado ==> getStr: ");

 Serial.println(getStr);

 delay(500);

 String messageBody = "";

 while (EspSerial.available())

 {

 String line = EspSerial.readStringUntil('\n');

 if (line.length() == 1)

 { *//actual content starts after empty line (that has 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";

 }

}

Преподавание основ алгоритмизации через программирование микроконтроллеров - это инновационный и эффективный подход, который делает обучение более занимательным, практичным и мотивирующим. Он позволяет студентам быстро усвоить основы программирования и применить их на практике, создавая собственные устройства и решения. Такой подход готовит учеников к жизни в цифровом мире и открывает для них новые возможности в будущем.