**7.Basic programs analog digital in and out + interrupt + timer + serial co**

A. analog input

const int buttonPin = 2; // Pin connected to the button

const int ledPin = 13; // Pin connected to the LED

void setup() {

pinMode(buttonPin, INPUT);

pinMode(ledPin, OUTPUT);

}

void loop() {

int buttonState = digitalRead(buttonPin);

if (buttonState == HIGH) {

digitalWrite(ledPin, HIGH);

}

else

{

digitalWrite(ledPin, LOW);

}

}

B. analog output

const int ledPin = 9;

void setup() {

pinMode(ledPin, OUTPUT);

}

void loop() {

for (int i = 0; i <= 255; i++) {

analogWrite(ledPin, i);

delay(10);

}

for (int i = 255; i >= 0; i--) {

analogWrite(ledPin, i);

delay(10);

}

}

C. Digital Input

const int buttonPin = 2; // Pin connected to the button

const int ledPin = 13; // Pin connected to the LED

void setup() {

pinMode(buttonPin, INPUT);

pinMode(ledPin, OUTPUT);

}

void loop() {

int buttonState = digitalRead(buttonPin);

if (buttonState == HIGH) {

digitalWrite(ledPin, HIGH);

} else {

digitalWrite(ledPin, LOW);

}

}

D. Digital output

const int ledPin = 13;

void setup() {

pinMode(ledPin, OUTPUT);

}

void loop() {

digitalWrite(ledPin, HIGH);

delay(1000); // Wait for a second

digitalWrite(ledPin, LOW);

delay(1000); // Wait for a second

}

E. Interrupt

const int buttonPin = 2;

const int ledPin = 13;

volatile bool ledState = LOW;

void setup() {

pinMode(buttonPin, INPUT);

pinMode(ledPin, OUTPUT);

attachInterrupt(digitalPinToInterrupt(buttonPin), toggleLED, CHANGE);

}

void loop() {

digitalWrite(ledPin, ledState);

}

void toggleLED() {

ledState = !ledState;

}

F.Timer

#include <TimerOne.h>

const int ledPin = 13;

void setup() {

pinMode(ledPin, OUTPUT);

Timer1.initialize(1000000); // Set timer for 1 second

Timer1.attachInterrupt(blinkLED);

}

void loop() {

// Main loop does nothing, blinkLED() is called by the timer interrupt

}

void blinkLED() {

static bool state = LOW;

state = !state;

digitalWrite(ledPin, state);

}

G. serial communication

const int sensorPin = A0;

void setup() {

Serial.begin(9600);

}

void loop() {

int sensorValue = analogRead(sensorPin);

Serial.print("Sensor Value: ");

Serial.println(sensorValue);

delay(500); // Delay for readability

}

**Project ideas and implementation resources**

Embedded Web Tech in Traffic Monitoring System

The principal objective of this project is to create software & hardware that can streamline audio data with the aid of USB 2.0. This is an XMOS & USB 2.0 based project design. With the aid of this project, we have brought forward reality in the form of a completely new innovative product.

This USB audio solution brings into play high-speed USB 2.0 with 480mb/s of audio data which delivers 24-bit audio. Approximately 192 KHz of sample frequency is received along with 2-40 audio channels. The extremely supple XMOS machines permit you to custom your solution to the precise combination of interfaces and digital procedures for your finished product.

This USB Audio solution 2.0 brings into use the XS1-L1 machine to make an available high quality digital audio connection for pro-audio and customer applications. The structure of the project also comprises of support for the USB Audio 1.0.

#### Mobile Operated SCADA for Industries Embedded System Project

#### The principal objective of this Mobile operated SCADA project is to formulate a system which can keep control over the working of a number of industrial machines at a time and to facilitate the remote usage of the equipment mobile technology is used. The Mobile operated Scada project is designed with equipment like- Microcontroller, smoke sensor, DTMF decoder, buzzer, a GSM mobile, temperature sensor to get an efficient control over the boiler division & packaging division.

#### This embedded project brings into play the DTMF decoder to connect GSM mobile with the microcontroller. The temperature sensor helps to sense the temperature in the boiler division & hence can be controlled. At the time of the fire, the smoke sensor helps & alerts by the buzzer. This embedded project is discovered in industrial units where control over processes is required distantly with some safety and validation.

#### Automation of Cars Using Embedded Systems Technology

#### In this project of automation cars, we have a receiver & a transmitter, the receiver collects all the signals from the transmitter. We can incorporate the system of automation car with a microcontroller or a microprocessor as per the need. The finger impression can be scanned and can be used to lock & unlock the car. A navigator can also be integrated into the system to give direction & location of the car.

#### The directions are provided to the car with the help of GPS technology. Airbags in the automation cars too can work on the embedded system & can save the driver at the time of the accident. A brake system monitoring device has to be added to get all the information regarding any disturbance caused in the brake system. A sensor to is incorporated in the car’s system which will automatically apply brakes or lower the speed when any obstacle is there.

#### Transients Control for Home Appliances Project

#### The key objective of this transients Control project is to intend a gadget which can control the transients that take place all through domestic applications. This transients Control project functions with a microcontroller & embedded system. A microcontroller is the integral or main part of the system. This system works on the commands entered. The programming language “C” is used in this transients Control project which is written into a microcontroller.

#### Sensors are brought into use to identify variations in the factors & conveys the message to analog to digital converter & the message converted is later conveyed to the microcontroller. The microcontroller compared the message with the values written in the program & if the values do not match any of the factors, the microcontroller will release command and thereby switching OFF the appliance.

#### Comparison across boards and code for same implementation

**1. Architecture**

STM32:

Architecture: ARM Cortex-M (e.g., Cortex-M0, M3, M4, M7, etc.)

Clock Speed: Up to 480 MHz (depending on the specific STM32 series)

Flash Memory: Up to 2 MB

SRAM: Up to 1 MB

Peripherals: Rich set of peripherals including USB, Ethernet, CAN, ADC, DAC, etc.

MSP430:

Architecture: 16-bit RISC

Clock Speed: Up to 25 MHz

Flash Memory: Up to 512 KB

SRAM: Up to 66 KB

Peripherals: Includes ADC, DAC, UART, SPI, I2C, and low-power modes.

Arduino Uno:

Architecture: 8-bit AVR (ATmega328P)

Clock Speed: 16 MHz

Flash Memory: 32 KB

SRAM: 2 KB

Peripherals: Digital I/O, PWM, UART, SPI, I2C, ADC.

8051:

Architecture: 8-bit CISC

Clock Speed: Typically 12 MHz (varies with different manufacturers)

Flash Memory: Up to 64 KB

SRAM: Typically up to 256 bytes (varies with different manufacturers)

Peripherals: Basic I/O, UART, timers, sometimes ADC and PWM.

**2. Development Environment**

STM32:

IDEs: STM32CubeIDE, Keil uVision, IAR Embedded Workbench, etc.

Tools: STM32CubeMX for peripheral and clock configuration.

Libraries: HAL (Hardware Abstraction Layer), LL (Low Layer), CMSIS.

MSP430:

IDEs: Code Composer Studio (CCS), IAR Embedded Workbench.

Tools: Grace for peripheral configuration.

Libraries: Driver Library, MSP430Ware.

Arduino Uno:

IDE: Arduino IDE (simple and beginner-friendly)

Libraries: Extensive collection of user-friendly libraries.

Tools: Arduino core libraries and community-contributed libraries.

8051:

IDEs: Keil uVision, SDCC (Small Device C Compiler).

Libraries: Vendor-specific and third-party libraries.

**3. Power Consumption**

STM32:

Power Consumption: Varies widely; ultra-low-power series (STM32L) can go down to a few µA in sleep modes.

Suitable for: Power-sensitive applications, battery-operated devices.

MSP430:

Power Consumption: Very low power, can operate in the range of nA in ultra-low-power modes.

Suitable for: Energy-efficient applications, wearable devices, and battery-operated projects.

Arduino Uno:

Power Consumption: Moderate; around 50 mA at 5V.

Suitable for: General-purpose applications, education, and prototyping.\

8051:

Power Consumption: Generally low, but depends on the specific variant.

Suitable for: Simple embedded systems and low-power applications.

**4. Performance**

STM32:

High performance due to ARM Cortex-M cores.

Suitable for: Real-time applications, complex processing tasks, and applications requiring high computational power.

MSP430:

Moderate performance with a focus on low power.

Suitable for: Low-power applications, simple control tasks.

Arduino Uno:

Basic performance; suitable for simple tasks.

Suitable for: Learning, simple DIY projects, and basic control tasks.

8051:

Basic performance with CISC architecture.

Suitable for: Legacy applications, simple control tasks.

**5. Ease of Use**

STM32:

Steeper learning curve, especially for beginners.

Requires familiarity with ARM Cortex-M architecture and HAL libraries.

MSP430:

Moderate learning curve.

Well-documented with good community support.

Arduino Uno:

Very easy to use, designed for beginners.

Extensive community support and simple programming environment.

8051:

Steeper learning curve due to CISC architecture and older toolchains.

Suitable for users familiar with 8-bit microcontrollers and low-level programming.

6. Typical Applications

STM32:

Advanced robotics, industrial automation, real-time systems, IoT devices, and multimedia applications.

MSP430:

Wearables, medical devices, energy harvesting applications, and other low-power projects.

Arduino Uno:

Educational projects, hobbyist electronics, DIY home automation, and simple prototyping.

8051:

Legacy systems, simple embedded tasks, industrial controllers, and educational purposes.

Summary

STM32: High performance, versatile, suitable for advanced applications, but with a steeper learning curve.

MSP430: Focuses on low power consumption, suitable for energy-efficient applications, moderate learning curve.

Arduino Uno: Beginner-friendly, easy to use, suitable for simple and educational projects.

8051: Legacy microcontroller with basic capabilities, suitable for simple control tasks and users familiar with older architectures.