Embedded Programming for Beginners

Implementing an embedded application using Arduino

Session 5

Course Goals

- Objectives:
  - Understand the debugging process.
  - Learn how to use USART to debug.

Digital Debugging

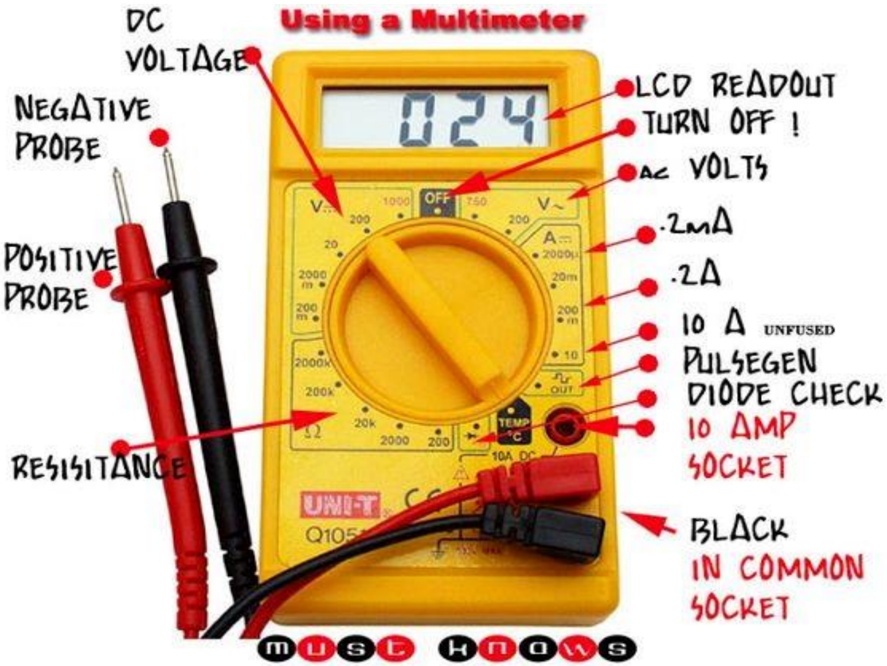
- Lack of IDEs or debuggers
- Error printing might not be an option
- If remote debugger support is available, it might lack specific support
- Lauterbach requires things like: Practice Scripting Language
- Hardware invasive behavior might be required
- Hardware might provide spurious errors

- However, we still must compare what is the expected vs unexpected behavior

Available Instruments

- LED debugging
- USART/Bluetooth etc. messages
- Advanced debuggers like JTAG
- Loop backing

- Multimeters
- Oscilloscopes
- Logic analyzers
- Protocol analyzers
- JTAG debuggers



0 MEASURE VOLTAGE IN PARALLEL WITH LOAD

0 MEASURE CURRENT IN SERIES WITH LOAD

0 MEASURE RESISTANCE IN A DEAD CIRCUIT

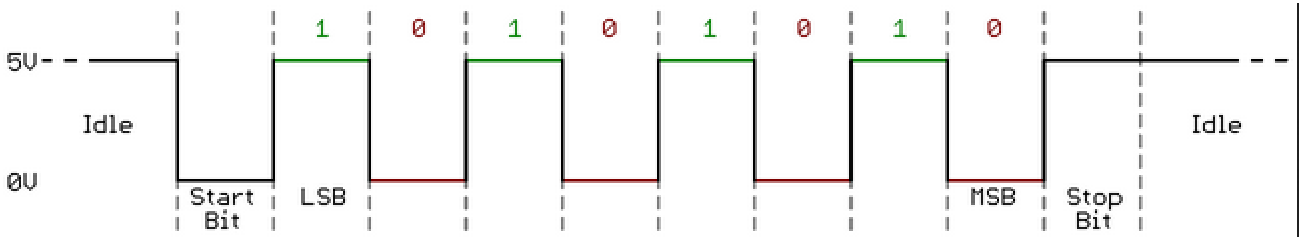
SET DIAL TO HIGH RANGE AND COME DOWN

Troubleshooting Flow

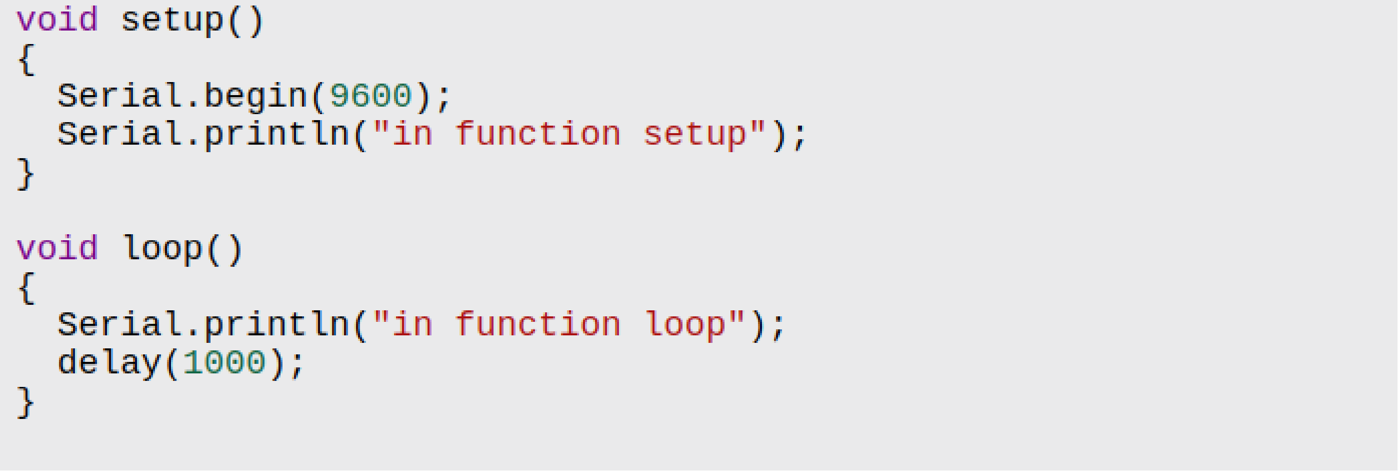
- Double check datasheet, diagram
- Double check registers configuration
- Check if peripherals are connected to pins
- Can we debug with printing messages
- Do we have an ethernet stack
- Do we have USART support
- Can we use LEDs for debugging
- Isolate the problem in smallest reproducible form

USART Serial Interface

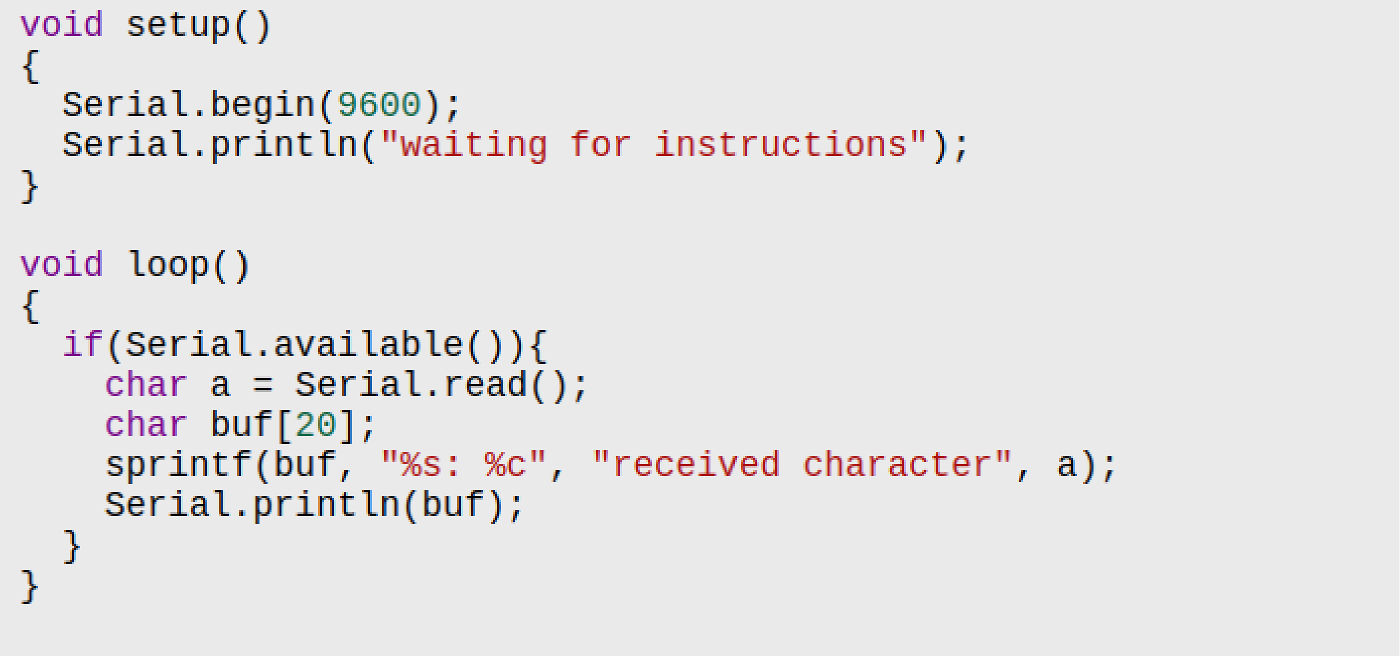
- Universal Synchronous-Asynchronous Receiver/Transmitter
- Full Duplex communication
  - Transmission line: Tx
  - Reception line: Rx
- Start bit
- Parity bit
- 1-2 stop bits



Arduino USART Demo

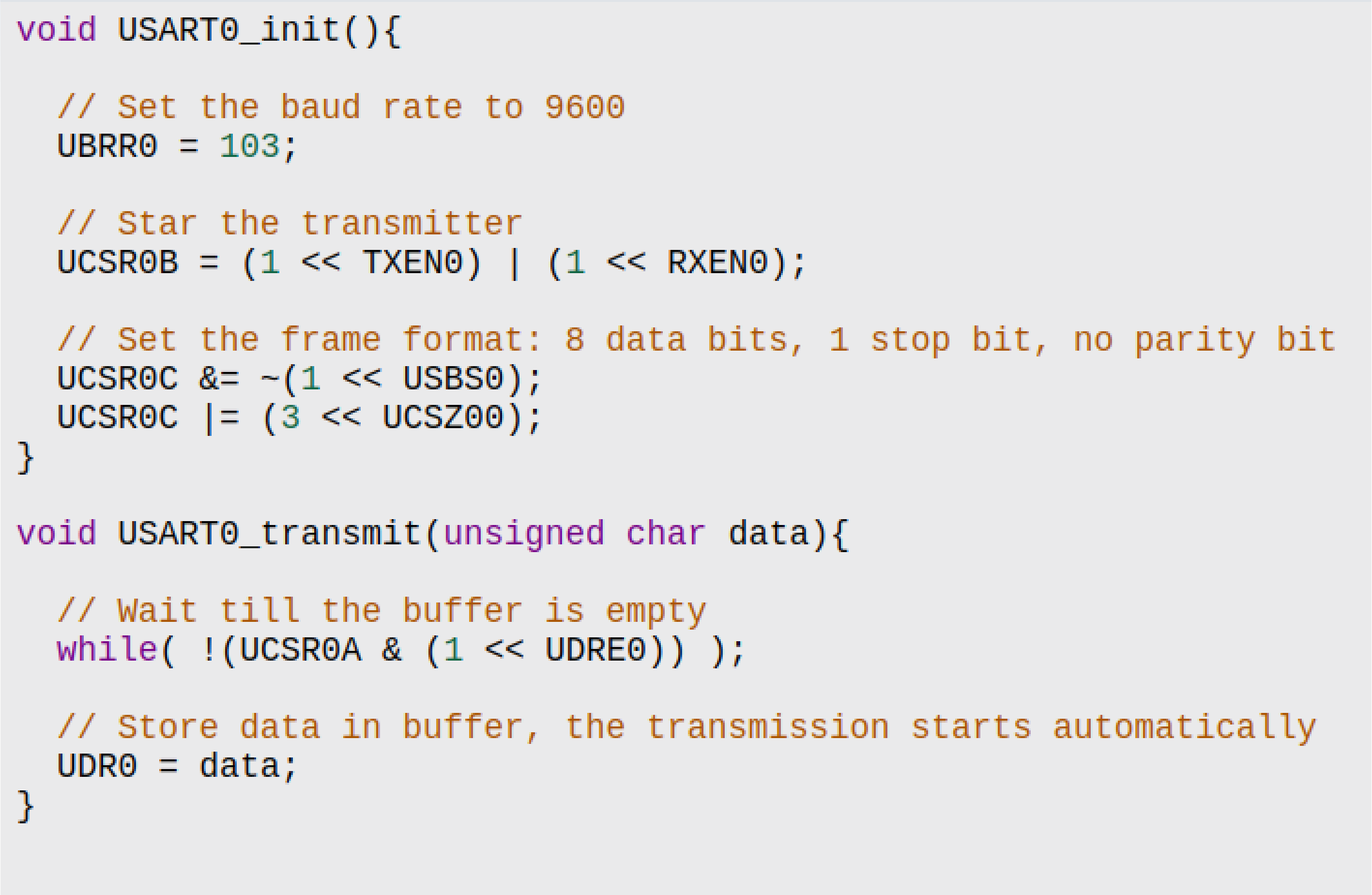


```
void setup()
{
  Serial.begin(9600);
  Serial.println("in function setup");
}

void loop()
{
  Serial.println("in function loop");
  delay(1000);
}
```



```
void setup()
{
  Serial.begin(9600);
  Serial.println("waiting for instructions");
}

void loop()
{
  if(Serial.available()){
    char a = Serial.read();
    char buf[20];
    sprintf(buf, "%s: %c", "received character", a);
    Serial.println(buf);
  }
}
```

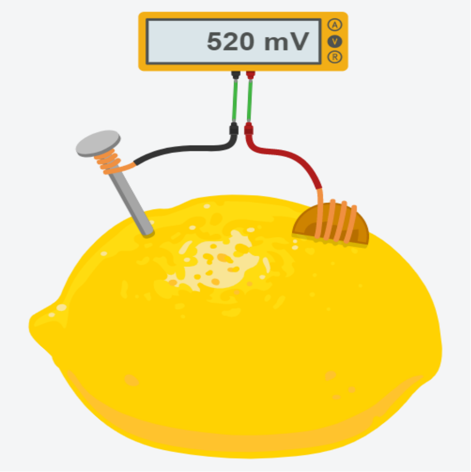
USART Demo



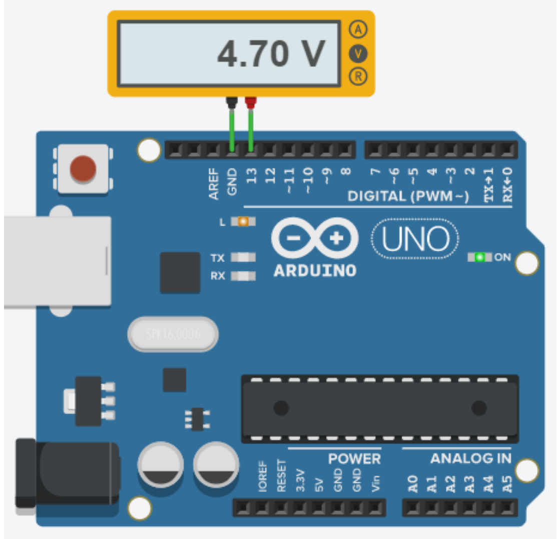
```
void USART0_init(){

  // Set the baud rate to 9600
  UBRR0 = 103;

  // Star the transmitter
  UCSR0B = (1 << TXEN0) | (1 << RXEN0);

  // Set the frame format: 8 data bits, 1 stop bit, no parity bit
  UCSR0C &= ~(1 << USBS0);
  UCSR0C |= (3 << UCSZ00);
}

void USART0_transmit(unsigned char data){

  // Wait till the buffer is empty
  while( !(UCSR0A & (1 << UDRE0)) );

  // Store data in buffer, the transmission starts automatically
  UDR0 = data;
}
```

Hands-on exercise: Arduino debugging

- Try using a voltmeter with a lime

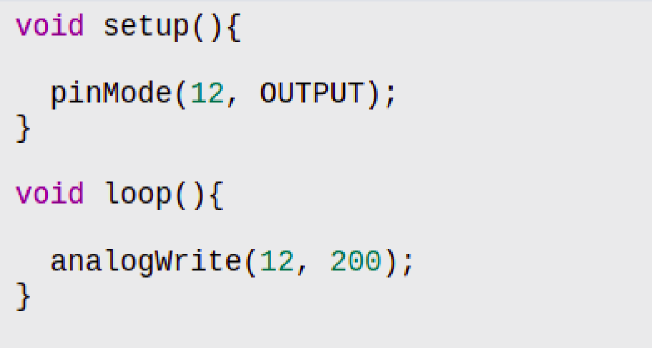


- Try using a voltmeter on a blinking LED

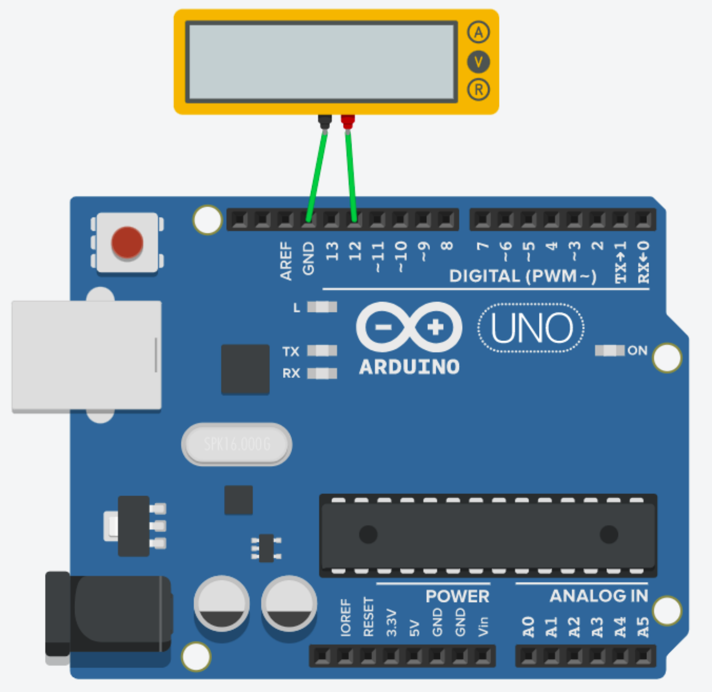


Hands-on exercise: Arduino debugging

- Debug analogWrite() setup to get 2V output

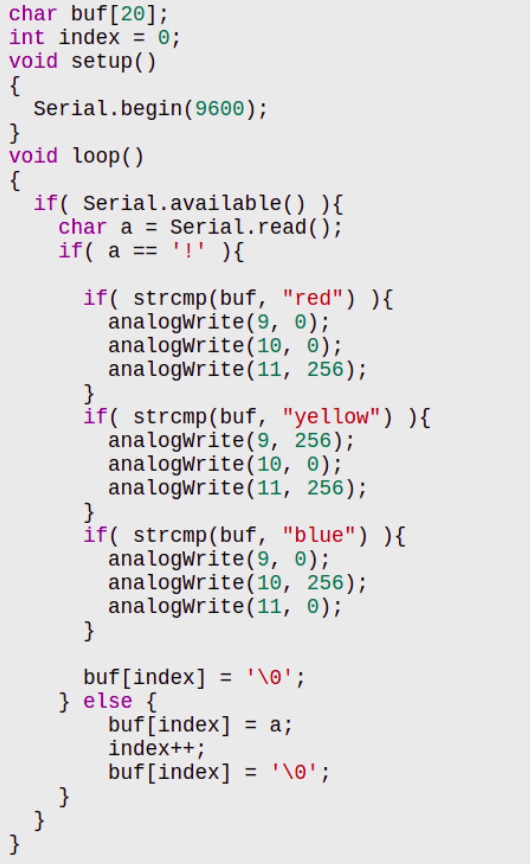


```
void setup(){

  pinMode(12, OUTPUT);
}

void loop(){

  analogWrite(12, 200);
}
```

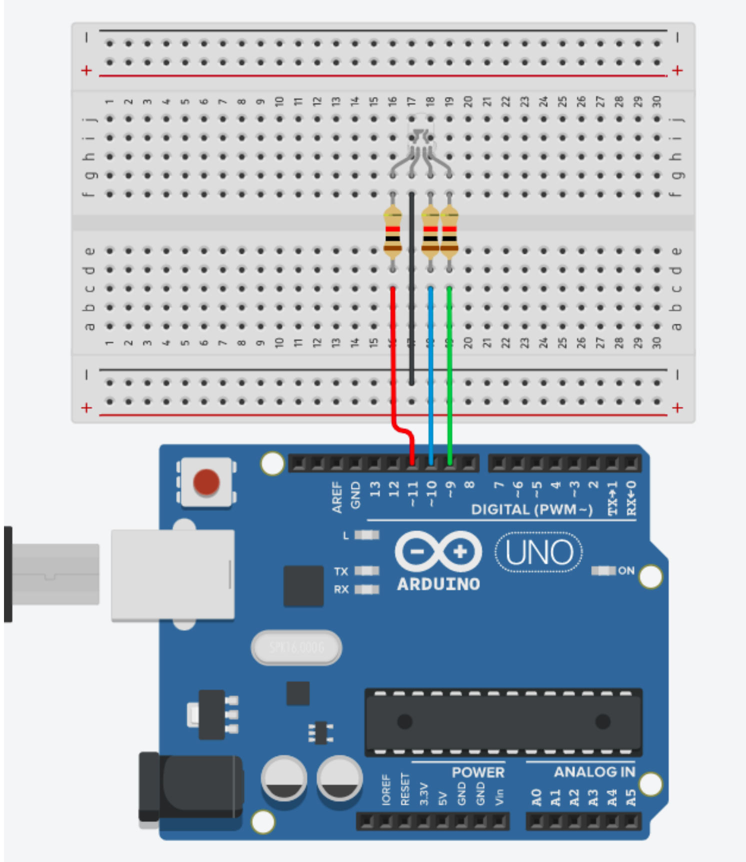


Hands-on exercise: Arduino debugging

- Debug the code and the circuit below to change the color of the RGB LED according to one of the messages sent: "red!", "yellow!" and "blue!".

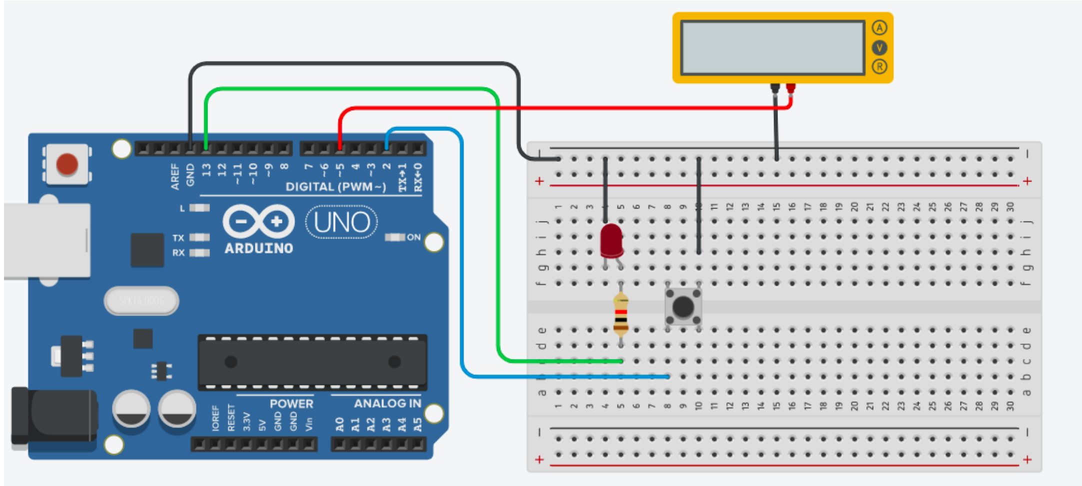


```
char buf[20];
int index = 0;
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  if( Serial.available() ){
    char a = Serial.read();
    if( a == '!' ){

      if( strcmp(buf, "red") ){
        analogWrite(9, 0);
        analogWrite(10, 0);
        analogWrite(11, 256);
      }
      if( strcmp(buf, "yellow") ){
        analogWrite(9, 256);
        analogWrite(10, 0);
        analogWrite(11, 256);
      }
      if( strcmp(buf, "blue") ){
        analogWrite(9, 0);
        analogWrite(10, 256);
        analogWrite(11, 0);
      }

      buf[index] = '\0';
    } else {
        buf[index] = a;
        index++;
        buf[index] = '\0';
    }
  }
}
```



Hands-on exercise: USART example

- Use serial console to send following commands:
  - “On” - lights an LED
  - “Off” - turns off an LED
  - “Blink” - blink an LED
  - “Get” - displays through the serial interface the status of a pressed button
  - “Analog” followed by a value - sets a voltage on a pin



Closing remarks

- USART
- Full-duplex
- Multimeters
- Oscilloscopes
- Logic analyzers
- Protocol analyzers
- JTAG debuggers