

## Introduction to Arduino IDE and getting started with the ESP32 microcontroller

### Part 5: Performing a calculation

Dr Ian Grout  
 Department of Electronic and Computer Engineering  
 Faculty of Science and Engineering  
 University of Limerick  
 Limerick, V94 T9PX  
 Ireland

Email: Ian.Grout@ul.ie



## Introduction

- In this part, the following activity will be:
  - Performing a calculation within a *calculation function* using values received from the serial port and transmitting the results back to the PC. Student exercise to modify the walkthrough example developed in part 4.
- A string will be sent to the microcontroller from the PC that will give the values as *float* type numbers required to calculate a value where:
 

• Input value	Call the variable <i>x</i> and its type will be <i>float</i>
• Gain	Call the variable <i>a</i> and its type will be <i>float</i>
• Offset	Call the variable <i>b</i> and its type will be <i>float</i>
• Output value	Call the variable <i>y</i> and its type will be <i>float</i>
- To perform the calculation:

$$y = ax + b$$



## How the system works

### Step 3

User enters the sketch code and uploads the compiled code to the microcontroller.

### Step 4

User opens the Serial Monitor and sends a string to the microcontroller.

### Step 6

User reads the strings received from the microcontroller.

### Step 2

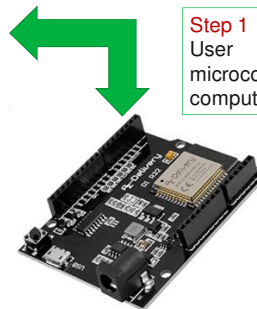
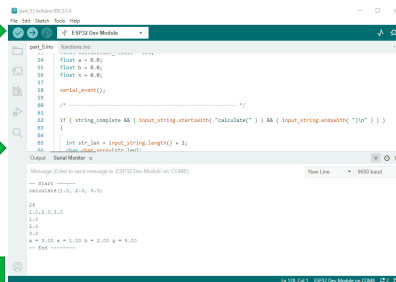
User creates a new, or opens an existing, Arduino sketch, selects the Arduino board and COM port to use.

### Step 5

The microcontroller receives the string from the user, performs the calculation and sends information strings back to the user.

### Step 1

User connects the microcontroller board to the computer



## The calculation

- Use the provided Arduino sketch **part\_5** and complete the calculation function in order to calculate the values for **y**.
- Verify the code by running it on the microcontroller and send different values for **a**, **b**, and **x**.

```
/* -----
 * float calculation( float a, float b, float x )
 * -----
 */

float calculation( float a, float b, float x )
{
    float y = 0.0;

    // Calculation code to go here

    return y;
}
```

$$y = ax + b$$

```
part_5 - Functions / Arduino 1.8.15
File Edit Sketch Tools Help

Functions

28 }
29
30 /* -----
31 * float calculation( float a, float b, float x )
32 * -----
33 * float calculation( float a, float b, float x )
34
35
36
37 float y = 0.0;
38
39 // Calculation code to go here
40
41 return y;
42
43 }
44
45 /* -----
46 * End of file
47 -----
*/
```

## Extracting data from the received string (1)

- Advanced topic.
- Note the code and it's basic operation.
- Using Arduino language and C language code.
- Review, get it working, then investigate to understand.

```
serial_event();

if ( string_complete && ( input_string.startsWith( "calculate(" ) && ( input_string.endsWith( ")\n" ) ) )
{
    ...
} else if ( string_complete )
{
    ...
} else
{
    ...
}
```

```
serial_event();

if ( string_complete && ( input_string.startsWith( "calculate(" ) && ( input_string.endsWith(
"\n" ) ) )
{
    int str_len = input_string.length() + 1;
    char char_array[str_len];

    Serial.println( "Start -----" );
    Serial.println( input_string );
    Serial.println( str_len );
    input_string.replace( "calculate(", "" );
    input_string.replace( ")", "" );
    input_string.replace( "\n", "" );
    Serial.println( input_string );

    input_string.toCharArray( char_array, str_len );
    token_ptr = strtok( char_array, "," );
    token_counter = 0;

    while ( token_ptr != NULL )
    {
        Serial.println( token_ptr );
        if ( token_counter==0 )
        {
            a = atof( token_ptr );
        }
        else if ( token_counter==1 )
        {
            b = atof( token_ptr );
        }
        else if ( token_counter==2 )
        {
            x = atof( token_ptr );
        }
        else
        {
            token_counter = token_counter + 1;
            token_ptr = strtok( NULL, "," );
        }
    }

    calculation_result = calculation( a, b, x );

    Serial.println( "x = " );
    Serial.println( x );
    Serial.println( "a = " );
    Serial.println( a );
    Serial.println( "b = " );
    Serial.println( b );
    Serial.println( "y = " );
    Serial.println( calculation_result );
    Serial.println( "End -----" );

    digitalWrite( LED_BUILTIN, digitalRead( LED_BUILTIN ) );
    input_string = "";
    string_complete = false;
} else if ( string_complete )
{
    Serial.println( "Incorrect value received" );
    Serial.println( input_string );
    input_string = "";
    string_complete = false;
}
else
{
    ...
}
```

## Extracting data from the received string (2)

Example string to send to the microcontroller  
(and includes a newline character)

calculate(1.0, 2.0, 3.0)

Last character  
sent from the  
computer

First character  
received by the  
microcontroller

\n ) 0 . 3 , 0 . 2 , 0 . 1 ( e t a l u c l a c

Computer

Microcontroller

```

if ( string_complete && ( input_string.startsWith( "calculate(" ) && ( input_string.endsWith( ")\n" ) ) )
{
    int str_len = input_string.length() + 1;
    char char_array[str_len];

    input_string.replace( "calculate(", " " );
    input_string.replace( " ", " " );
    input_string.replace( ")\n", " " );

    input_string.toCharArray( char_array, str_len );
    token_ptr = strtok( char_array, "," );
    token_counter = 0;

    while ( token_ptr !=NULL )
    {
        if ( token_counter==0 )
        {
            a = atof( token_ptr );

        } else if ( token_counter==1 )
        {
            b = atof( token_ptr );

        } else if ( token_counter==2 )
        {
            x = atof( token_ptr );
        } else
        {
        }
        token_counter = token_counter + 1;
        token_ptr = strtok( NULL, "," );
    }

    calculation_result = calculation( a, b, x );

    input_string = "";
    string_complete = false;
}

```

## Extracting data from the received string (3)



```

if ( string_complete && ( input_string.startsWith( "calculate(" ) && ( input_string.endsWith( ")\n" ) ) )
{

```

```

    int str_len = input_string.length() + 1;
    char char_array[str_len];

    input_string.replace( "calculate(", " " );
    input_string.replace( " ", " " );
    input_string.replace( ")\n", " " );

    input_string.toCharArray( char_array, str_len );
    token_ptr = strtok( char_array, "," );
    token_counter = 0;

```

```

while ( token_ptr !=NULL )
{
    if ( token_counter==0 )
    {
        a = atof( token_ptr );

    } else if ( token_counter==1 )
    {
        b = atof( token_ptr );

    } else if ( token_counter==2 )
    {
        x = atof( token_ptr );
    } else
    {
    }
    token_counter = token_counter + 1;
    token_ptr = strtok( NULL, "," );
}

```

```

    calculation_result = calculation( a, b, x );

```

```

    input_string = "";
    string_complete = false;

```

```

}

```

Declare local variables

Extract characters from string

Perform calculation

Reset string related variables

## Extracting data from the received string (4)



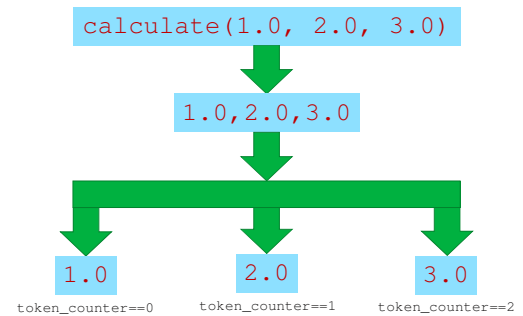
```
int str_len = input_string.length() + 1;
char char_array[str_len];
```

```
input_string.replace( "calculate(", " " );
input_string.replace( " ", " " );
input_string.replace( "\\n", " " );

input_string.toCharArray( char_array, str_len );
token_ptr = strtok( char_array, " " );
token_counter = 0;
```

```
while ( token_ptr !=NULL )
{
    if ( token_counter==0 )
    {
        a = atof( token_ptr );
    } else if ( token_counter==1 )
    {
        b = atof( token_ptr );
    } else if ( token_counter==2 )
    {
        x = atof( token_ptr );
    } else
    {
        token_counter = token_counter + 1;
        token_ptr = strtok( NULL, " " );
    }
}
```

## Extracting data from the received string (5)

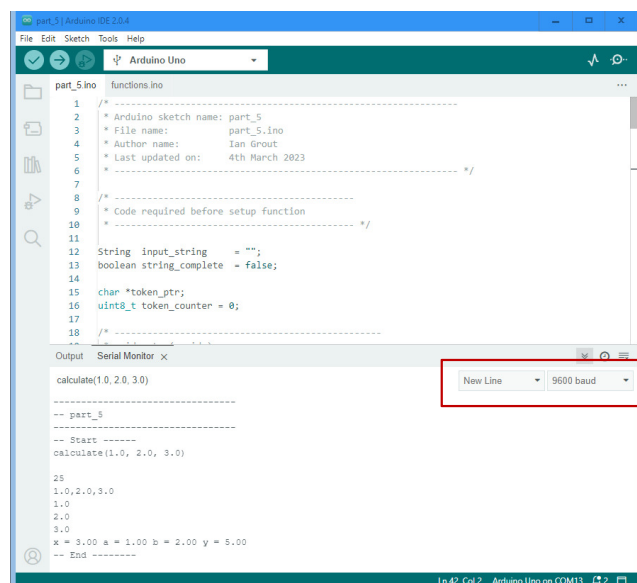


## Exercise

- Obtain the **part\_5** Arduino sketch and complete the calculation function.
- Use the Arduino Serial Monitor to send values to the microcontroller by entering the following string:

```
calculate(1.0, 2.0, 3.0)
```

- Vary the numbers to change the values for **a**, **b**, and **x**.
- Watch the **video part\_5\_video.mp4** to see the completed sketch in operation.



## Python script to replace the Arduino IDE Serial Monitor

```
import time
import serial

com_port = 'COM8'

def main():

    ser = serial.Serial(com_port, timeout=5)
    ser.baudrate = 9600
    ser.flush()
    time.sleep(5)
    print(ser.name)

    for i in range(0, 3):
        line = ser.readline().decode('latin-1')[:-1]
        print(line)

    value_to_send = 'calculate(1.0, 2.0, 3.0)\n'

    print(value_to_send)
    ser.write(value_to_send.encode())

    time.sleep(1)

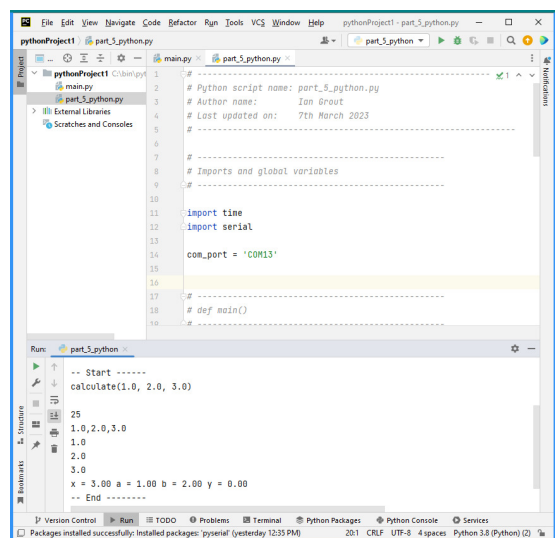
    for i in range(0, 10):
        line = ser.readline().decode('latin-1')[:-1]
        print(line)

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

- The Arduino IDE Serial Monitor is useful for initial prototyping and debugging the design code.
- For more advanced work, other software languages and tools can be used.
- For example, using Python to access the serial port as shown in the example to the left.
- This example uses **pySerial** to access the serial port. This is the same COM PORT as set in the Arduino IDE.
- In the code, COM8 is used on a Windows platform.

## Python script to replace the Arduino IDE Serial Monitor

- Python scripts can be created and run using different software tools.
- For example, the image to the right shows the Python script developed and using PyCharm Community Edition.
- The Python script is **part\_5\_python.py**.
- Watch the video **part\_5\_python\_video** to see Arduino IDE and PyCharm in use.



**Any questions?**



University of Limerick,  
Limerick, V94 T9PX,  
Ireland.  
Ollscoil Luimnigh,  
Luimneach,  
V94 T9PX, Éire.  
+353 (0) 61 202020

[ul.ie](http://ul.ie)