Introduction to Arduino IDE and getting started with the ESP32 microcontroller

# Part 4: Receiving strings from the computer and string manipulation to extract values from the string

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

UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

# Introduction

- Receiving strings from the computer and string manipulation to extract values from the string. Walk-through example. Send the extracted values back to the PC:

1. The microcontroller receiving and sending serial data using serial communications (UART).
2. The microcontroller receiving strings.
3. The microcontroller extracting values from a string.
4. The microcontroller formatting and transmitting data.
5. Walkthrough example using the Arduino IDE Serial Monitor and then in Python.

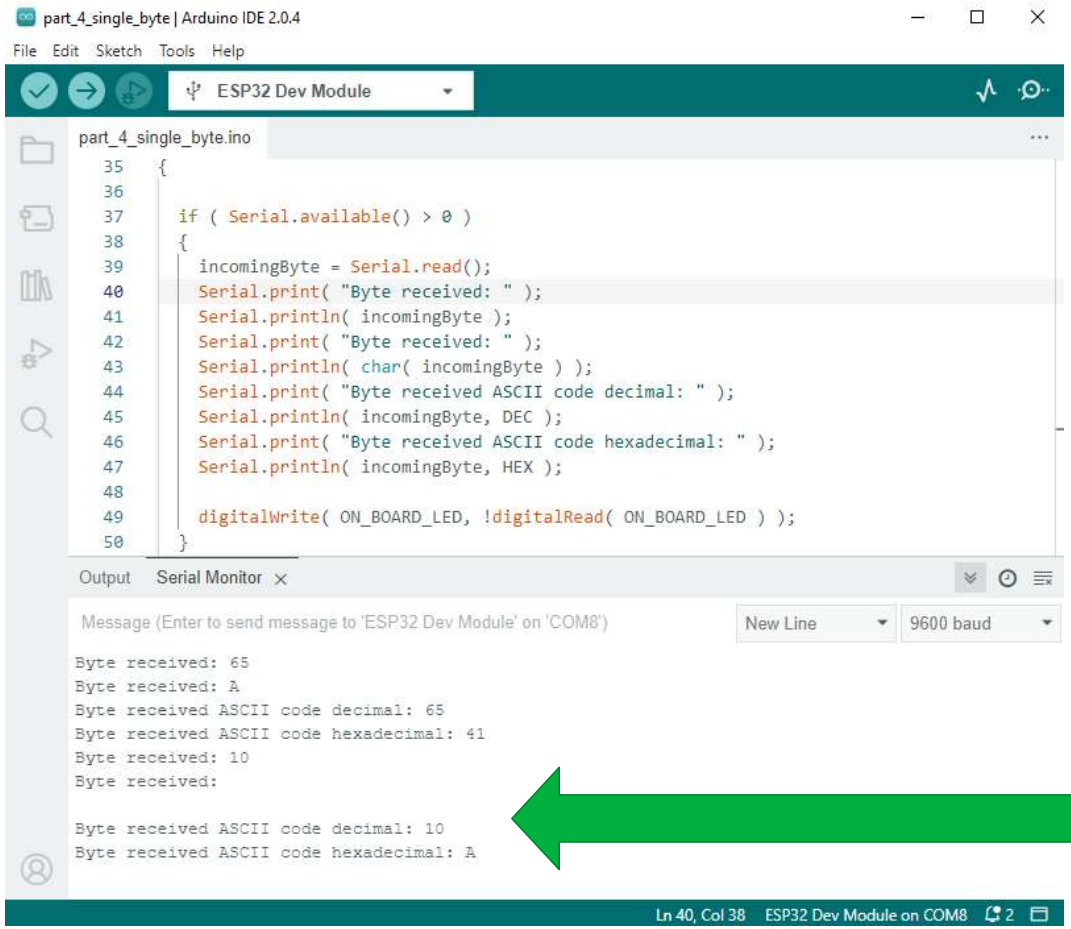
# Receiving and transmitting a single byte (1)

- Arduino serial communication function:
  - https://www.arduino.cc/reference/en/language/functions/communication/serial/
- The microcontroller UART set-up with a Baud rate of 9600.
- The UART is checked (polled) to see if a byte has been received. If received, the byte is read and transmitted back to the computer as the value and also as the ASCII character code.
- The code is in the Arduino Sketch part_4_single_byte.
- Watch the video part_4_single_byte_video.mp4 to see the code in operation. This code also toggles the on-board LED when a byte is received (the LED code is not shown in the code to the right).

```
int incomingByte = 0;


void setup( void )
{

  Serial.begin(9600);

}


void loop( void )
{

  if ( Serial.available() > 0 )
  {
    incomingByte = Serial.read();
    Serial.print( "Byte received: " );
    Serial.println( incomingByte );
    Serial.print( "Byte received ASCII code decimal: " );
    Serial.println( incomingByte, DEC );
    Serial.print( "Byte received ASCII code hexadecimal: " )
    Serial.println( incomingByte, HEX );
  }

}
```

UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

# Receiving and transmitting a single byte (2)




```
    Serial.print( "Byte received: " );
    Serial.println( incomingByte );
    Serial.print( "Byte received: " );
    Serial.println( char( incomingByte ) );
    Serial.print( "Byte received ASCII code decimal: " );
    Serial.println( incomingByte, DEC );
    Serial.print( "Byte received ASCII code hexadecimal: " );
    Serial.println( incomingByte, HEX );
```

UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

# ASCII code

- ASCII: American Standard Code for Information Interchange.
- ASCII is a 7-bit character set containing 128 characters.
- Extended ASCII is an 8-bit character set containing 256 characters.
- ASCII Table:
  - https://commons.wikimedia.org/wiki/File:ASCII-Table-wide.pdf

## ASCII TABLE

| Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | [NULL] | 32 | 20 | [SPACE] | 64 | 40 | @ | 96 | 60 | ` |
| 1 | 1 | [START OF HEADING] | 33 | 21 | ! | 65 | 41 | A | 97 | 61 | a |
| 2 | 2 | [START OF TEXT] | 34 | 22 | " | 66 | 42 | B | 98 | 62 | b |
| 3 | 3 | [END OF TEXT] | 35 | 23 | # | 67 | 43 | C | 99 | 63 | c |
| 4 | 4 | [END OF TRANSMISSION] | 36 | 24 | $ | 68 | 44 | D | 100 | 64 | d |
| 5 | 5 | [ENQUIRY] | 37 | 25 | % | 69 | 45 | E | 101 | 65 | e |
| 6 | 6 | [ACKNOWLEDGE] | 38 | 26 | & | 70 | 46 | F | 102 | 66 | f |
| 7 | 7 | [BELL] | 39 | 27 | ' | 71 | 47 | G | 103 | 67 | g |
| 8 | 8 | [BACKSPACE] | 40 | 28 | ( | 72 | 48 | H | 104 | 68 | h |
| 9 | 9 | [HORIZONTAL TAB] | 41 | 29 | ) | 73 | 49 | I | 105 | 69 | i |
| 10 | A | [LINE FEED] | 42 | 2A | * | 74 | 4A | J | 106 | 6A | j |
| 11 | B | [VERTICAL TAB] | 43 | 2B | + | 75 | 4B | K | 107 | 6B | k |
| 12 | C | [FORM FEED] | 44 | 2C | , | 76 | 4C | L | 108 | 6C | l |
| 13 | D | [CARRIAGE RETURN] | 45 | 2D | - | 77 | 4D | M | 109 | 6D | m |
| 14 | E | [SHIFT OUT] | 46 | 2E | . | 78 | 4E | N | 110 | 6E | n |
| 15 | F | [SHIFT IN] | 47 | 2F | / | 79 | 4F | O | 111 | 6F | o |
| 16 | 10 | [DATA LINK ESCAPE] | 48 | 30 | 0 | 80 | 50 | P | 112 | 70 | p |
| 17 | 11 | [DEVICE CONTROL 1] | 49 | 31 | 1 | 81 | 51 | Q | 113 | 71 | q |
| 18 | 12 | [DEVICE CONTROL 2] | 50 | 32 | 2 | 82 | 52 | R | 114 | 72 | r |
| 19 | 13 | [DEVICE CONTROL 3] | 51 | 33 | 3 | 83 | 53 | S | 115 | 73 | s |
| 20 | 14 | [DEVICE CONTROL 4] | 52 | 34 | 4 | 84 | 54 | T | 116 | 74 | t |
| 21 | 15 | [NEGATIVE ACKNOWLEDGE] | 53 | 35 | 5 | 85 | 55 | U | 117 | 75 | u |
| 22 | 16 | [SYNCHRONOUS IDLE] | 54 | 36 | 6 | 86 | 56 | V | 118 | 76 | v |
| 23 | 17 | [ENG OF TRANS. BLOCK] | 55 | 37 | 7 | 87 | 57 | W | 119 | 77 | w |
| 24 | 18 | [CANCEL] | 56 | 38 | 8 | 88 | 58 | X | 120 | 78 | x |
| 25 | 19 | [END OF MEDIUM] | 57 | 39 | 9 | 89 | 59 | Y | 121 | 79 | y |
| 26 | 1A | [SUBSTITUTE] | 58 | 3A | : | 90 | 5A | Z | 122 | 7A | z |
| 27 | 1B | [ESCAPE] | 59 | 3B | ; | 91 | 5B | [ | 123 | 7B | { |
| 28 | 1C | [FILE SEPARATOR] | 60 | 3C | < | 92 | 5C | \ | 124 | 7C | \| |
| 29 | 1D | [GROUP SEPARATOR] | 61 | 3D | = | 93 | 5D | ] | 125 | 7D | } |
| 30 | 1E | [RECORD SEPARATOR] | 62 | 3E | > | 94 | 5E | ^ | 126 | 7E | ~ |
| 31 | 1F | [UNIT SEPARATOR] | 63 | 3F | ? | 95 | 5F | _ | 127 | 7F | [DEL] |

| Decimal | Hex | Char |
|---|---|---|
| 64 | 40 | @ |
| 65 | 41 | A |
| 66 | 42 | B |
| 67 | 43 | C |

UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

# Receiving multiple bytes with a termination character (1)

- Serial communications will send data one byte at a time.
- Where multiple bytes are to be received by the microcontroller in “one go”, such as in a string of text, each byte in the string would be received and put into a character array variable within the code (a String in Arduino code).
- The microcontroller needs to know when the string ends, so a termination character, such as the end of line character ( \n ) will be used to identify the end of the string. For example:

This is a string\n

- T is transmitted first and \n is transmitted last.
- The Arduino Serial Monitor can insert the \n character or the code writer can insert this with code.

# Receiving multiple bytes with a termination character (2)

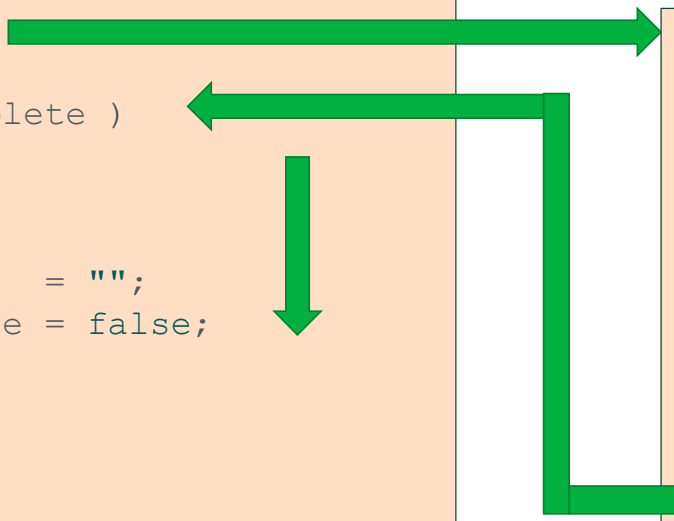
```
String  input_string     = "";
boolean string_complete  = false;
```

```
void loop( void )
{

  serial_event();

  if ( string_complete )
  {
    Serial.print( "String received: " );
    Serial.println( input_string );

    digitalWrite( ON_BOARD_LED, !digitalRead( ON_BOARD_LED ) );

    input_string    = "";
    string_complete = false;
  } else
  {
  }

}
```

# Receiving multiple bytes with a termination character (3)

```
void loop( void )
{

  serial_event();

  if ( string_complete )
  {
    …

    input_string    = "";
    string_complete = false;

  } else
  {
  }

}
```

```
void serial_event( void )
{

  while ( Serial.available() )
  {

    char in_char = ( char )Serial.read();
    input_string += in_char;

    if ( in_char == '\n' )
    {
      string_complete = true;
    }

  }

}
```



# Receiving multiple bytes with a termination character (4)

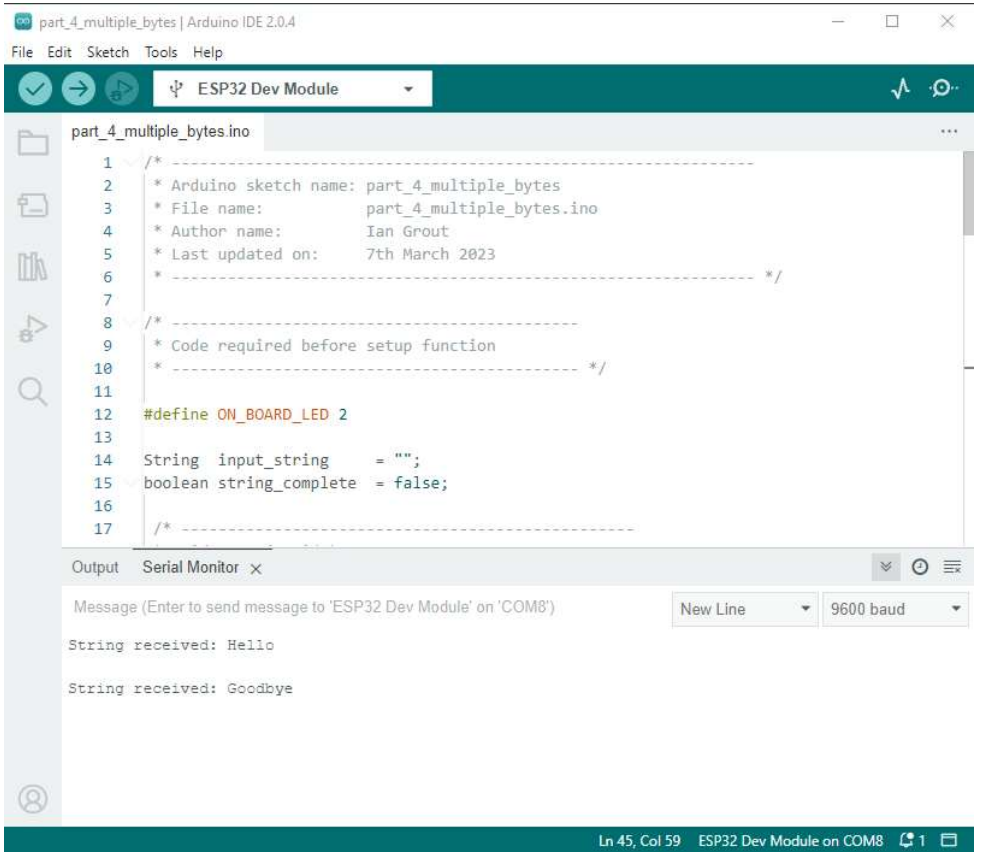
- Arduino serial communication function:
  - https://www.arduino.cc/reference/en/language/functions/communication/serial/

- The microcontroller UART set-up with a Baud rate of 9600.

- The UART is checked (polled) to see if a byte has been received. If received, the byte is read and transmitted back to the computer as the value and also as the ASCII character code.

- The code is in the Arduino Sketch part_4_multiple_bytes.

- Watch the video part_4_multiple_bytes_video.mp4 to see the code in operation. This code also toggles the on-board LED when a byte is received (the LED code is not shown in the code to the right).




UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

# Replacing the Arduino Serial Monitor with Python

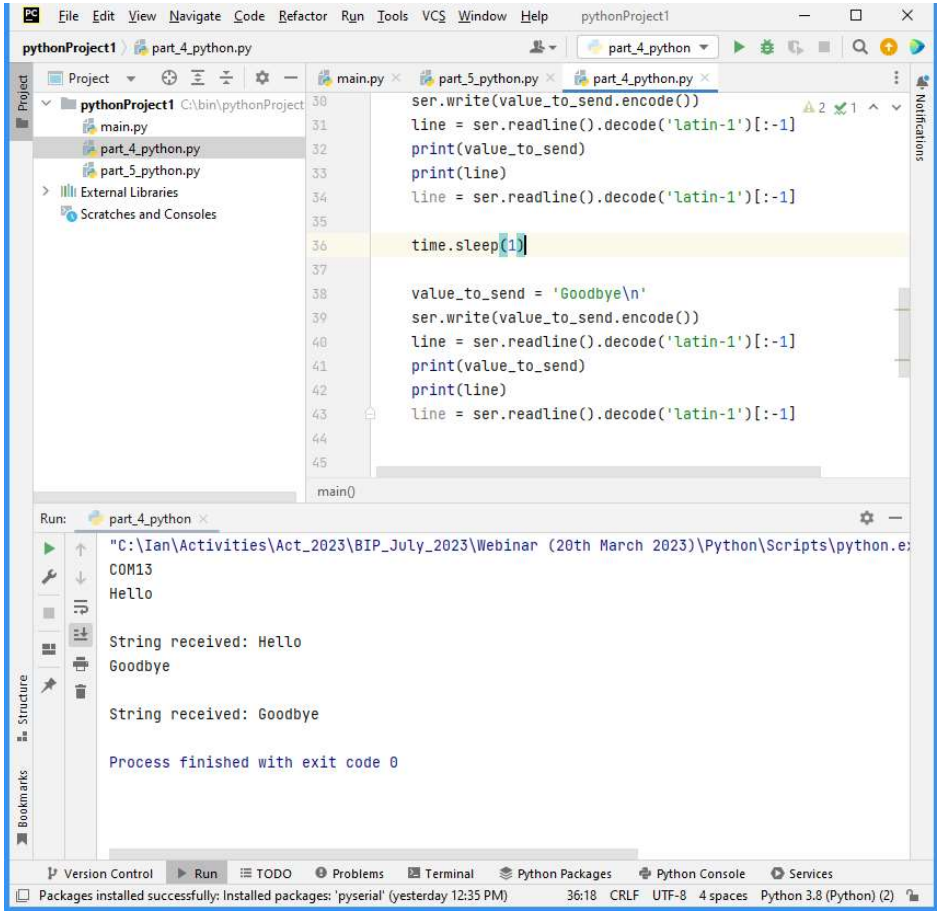
```
import time
import serial

com_port = 'COM8'


def main():

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

    value_to_send = 'Hello\n'
    ser.write(value_to_send.encode())
    line = ser.readline().decode('latin-1')[:-1]
    print(value_to_send)
    print(line)
    line = ser.readline().decode('latin-1')[:-1]

    time.sleep(1)

    value_to_send = 'Goodbye\n'
    ser.write(value_to_send.encode())
    line = ser.readline().decode('latin-1')[:-1]
    print(value_to_send)
    print(line)
    line = ser.readline().decode('latin-1')[:-1]


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. This should be replaced with the actual COM PORT number used.

UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

# Replacing the Arduino Serial Monitor with Python

- 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_4_python.py .
- Watch the video part_4_python_video to see Arduino IDE and PyCharm in use.




UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

# Any questions?

UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

University of Limerick,
Limerick, V94 T9PX,
Ireland.

Ollscoil Luimnigh,
Luimneach,
V94 T9PX, Éire.

+353 (0) 61 202020

ul.ie