

REYKJAVÍK UNIVERSITY

MECHATRONICS I

T-411-MECH



INDIVIDUAL ASSIGNMENT 5

14/09/2020

Students:

Giacomo Menchi

Teacher:

Joseph T. Foley

Contents

1 Tasks	1
1.1 Task 1	1
1.2 Task 2	1
1.3 Task 3	2
1.4 Task 4	3
1.5 Task 5	5
1.6 Task 6	6
1.7 Task 7	7
1.8 Task 8	7
1.9 Task 9	9
1.10 Task 10	9

1 Tasks

As usual, the code for this assignment is available at the following Github repo:

<https://shorturl.at/uEUY2>

1.1 Task 1

Task: Get started with Arduino and Arduino IDE. Open blink example and put your name in the comment.

Screenshot: This was quite the easy task, so just check the screenshot below.

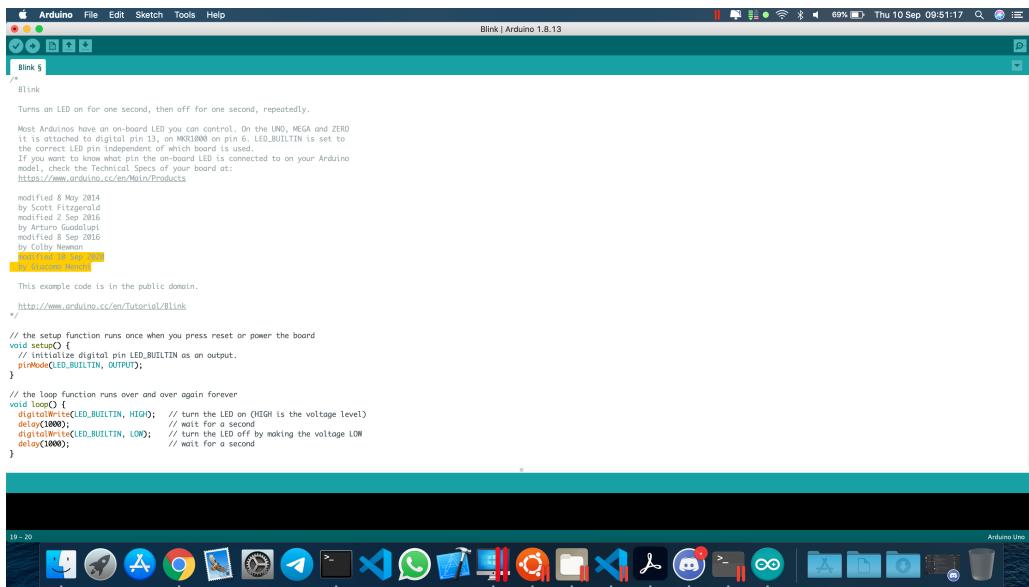


Figure 1: Blink example with my name on the comment.

1.2 Task 2

Task: Set Arduino Nano as the target board and set the com port to where the nano is. Blink a led on it with two different rates.

Video: Again, to complete this task I only needed to adjust the aforementioned settings and the Processor to ATmega328P (Old bootloader), and then edit some lines from the Blink sketch.

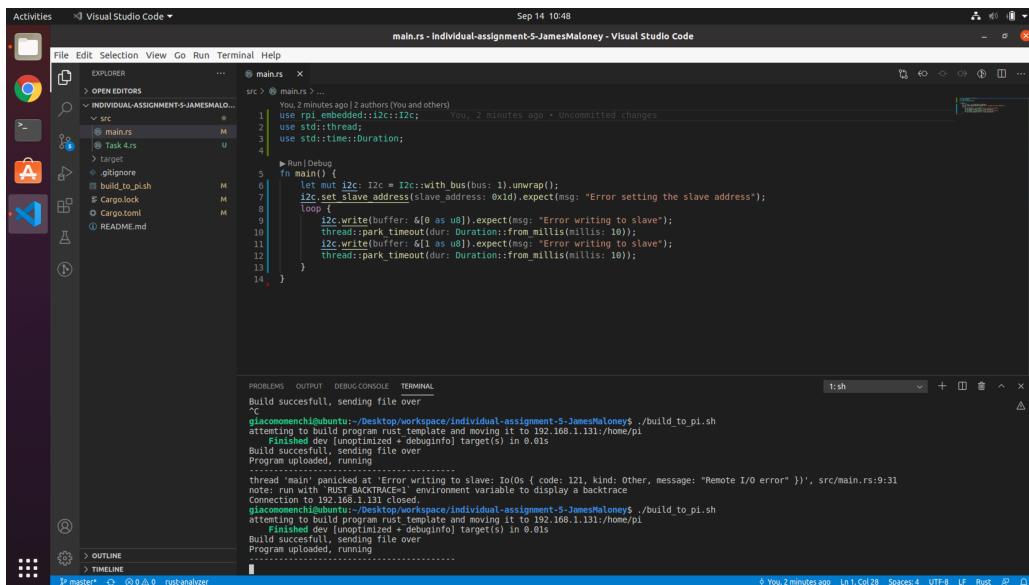
<https://shorturl.at/hCFT6>

1.3 Task 3

Task: Make echo code for I2C connection between the Arduino and the RPi so the Arduino is the slave device and the RPi is the Master.

NOTE; use Wire library for Arduino

Screenshot: As shown in the screenshots below, I just programmed the Raspberry Pi to send a "1" followed by a "0" to the Arduino in a looping (neverending) way, and the Arduino to receive those messages and print them on its serial console.



```

Activities <- Visual Studio Code
File Edit Selection View Go Run Terminal Help
OPEN EDITORS
INDIVIDUAL-ASSIGNMENT-5-JAMESMALONE...
src
main.rs M
Task 4.rs U
target
.githignore
build_to_pi.sh
Cargo.lock
Cargo.toml
README.md
main.rs - Individual-Assignment-5-JamesMaloney - Visual Studio Code
Sep 14 10:48
main.rs
You, 2 minutes ago | 2 authors (You and others)
1 use rpl_embedded::i2c;
2 use std::thread;
3 use std::time::Duration;
4
5 fn main() {
6     let mut i2c: I2c = I2c::with_bus(bus: 1).unwrap();
7     i2c.set_slave_address(slave_address: 0x0D).expect(msg: "Error setting the slave address");
8     loop {
9         i2c.write(buffer: &[0 as u8]).expect(msg: "Error writing to slave");
10        thread::park_timeout(dur: Duration::from_millis(millis: 10));
11        i2c.read(buffer: &mut [0 as u8]).expect(msg: "Error reading from slave");
12        thread::park_timeout(dur: Duration::from_millis(millis: 10));
13    }
14}

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Build successful, sending file over
`c
giacomenchi@ubuntu:~/Desktop/workspace/individual-assignment-5_JamesMaloney\$./build_to_pi.sh
attempting to build program rust template and moving it to 192.168.1.131:/home/pi
finished dev [unoptimized + debuginfo] target(s) in 0.01s
Build successful, sending file over
Program uploaded, running.....
thread 'main' panicked at 'Error writing to slave: Io{0 { code: 121, kind: Other, message: "Remote I/O error" }}', src/main.rs:9:31
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
Connection to 192.168.1.131 closed.
giacomenchi@ubuntu:~/Desktop/workspace/individual-assignment-5_JamesMaloney\$./build_to_pi.sh
attempting to build program rust template and moving it to 192.168.1.131:/home/pi
finished dev [unoptimized + debuginfo] target(s) in 0.01s
Build successful, sending file over
Program uploaded, running.....

Figure 2: Echo code for Raspberry Pi in Rust.

The screenshot shows the Arduino IDE interface. The top bar displays the title "I2C_Rasp" and the version "Arduino 1.8.13". The status bar at the bottom right shows the date and time as "Mon 14 Sep 10:48:19". The main area contains the code for the I2C communication sketch, which includes the header "#include <Wire.h>" and the setup/loop functions for reading from the Raspberry Pi's I2C address. The serial monitor window is open, connected to "/dev/cu.usbserial-1440" at 9600 baud, showing the received data from the Raspberry Pi. The bottom status bar indicates the sketch uses 3360 bytes of program storage space and 372 bytes of dynamic memory.

```
#include <Wire.h>
void setup() {
    Serial.begin(9600);
    Wire.begin();
    Wire.onReceive(receiveEvent);
}
void receiveEvent(int howMany) {
    while (Wire.available()) {
        int i = Wire.read();
        Serial.print(i);
    }
}
void loop() {
    delay(100);
}
```

Sketch uses 3360 bytes (10%) of program storage space. Maximum is 30720 bytes.
Global variables use 372 bytes (1%) of dynamic memory, leaving 1676 bytes for local variables. Maximum is 2048 bytes.

Figure 3: Echo code for Arduino in C.

1.4 Task 4

Task: Create code to receive and send any keyboard typed data to and from the I2C port on the Rpi in Rust.

Screenshot: This is the proof demonstrating the echo server code in Arduino (and Raspberry Pi). Whenever a string is sent to the Arduino, it's received, saved, and also sent back to the Raspberry Pi itself.

```

Activities > Visual Studio Code >
File Edit Selection View Go Run Terminal Help
OPEN EDITORS
src > main.rs > main.rs
main.rs - individual-assignment-5-JamesMaloney - Visual Studio Code
Sep 15 13:40
10     i2c.set_slave_address(slave_address: 0x10).expect(msg: "slave address failed");
11
12     let mut input: String;
13     let mut output: String;
14
15     loop {
16         input = String::from("");
17         output = String::from("");
18         print!("Enter a string: ");
19         let mut stdout = std::io::stdout();
20         stdout.write_all(b"\n").expect(msg: "Please enter a correct string");
21         let mut read_vector: Vec = vec![0; input.len()];
22         i2c.read_buffer(&mut read_vector).expect(msg: "Err during reading");
23         for character: u8 in read_vector.to_vec() {
24             output.push(character as char);
25         }
26     }
27     println!("Received string: {}", output);
28 }
29
30

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

giacommachi@ubuntu:~/Desktop/workspace/individual-assignment-5-JamesMaloney$ ./build_to_pi.sh
attempting to build program rust template and moving it to 192.168.1.131:/home/pi
Compiling rust template v0.1.0 (/home/giacommachi/Desktop/workspace/individual-assignment-5-JamesMaloney)
Finished dev [unoptimized + debugging] target(s) in 0.44s
Built binary at: /home/giacommachi/Desktop/workspace/individual-assignment-5-JamesMaloney/target/debug/rust-template
Program uploaded, running
-----
Enter a string: Giacomo has a new echo server
Received string: Giacomo has a new echo server
Enter a string: "Connection to 192.168.1.131 closed.
giacommachi@ubuntu:~/Desktop/workspace/individual-assignment-5-JamesMaloney$ 

```

You, a few seconds ago Ln 28, Col 25 Spaces: 4 UTF-8 LF Rust

Figure 4: Echo code for Raspberry Pi in Rust.

```

Arduino File Edit Sketch Tools Help
Task_4 | Arduino 1.8.13
Task_4
* Giacomo Menchi
* IA S
*/
#include <Wire.h>
String s = "+";
void setup() {
  Serial.begin(9600);
  Wire.begin(40);
  Wire.onReceive(receiveEvent);
  Wire.onRequest(requestEvent);
}
void receiveEvent(int howMany) {
  while(Wire.available() > 0) {
    char c = Wire.read();
    s+=c;
  }
}
void requestEvent() {
  Wire.write(s.c_str());
  s = "+";
}
void loop() {
  delay(1000);
}

```

Done uploading.
Sketch uses 4580 bytes (14%) of program storage space. Maximum is 30720 bytes.
Global variables use 388 bytes (1%) of dynamic memory, leaving 1660 bytes for local variables. Maximum is 2048 bytes.

Arduino Nano on /dev/cu.usbserial-1430

Figure 5: Echo code for Arduino in C.

1.5 Task 5

Task: Make the Arduino and the RPi communicate by sending numbers back and forth where the Rpi is adding 10 and the Arduino is adding 1 when they receive data and then sending it back. Print the incoming number on each device terminal.

Screenshot: Again, the screenshots below represent the implementation of the task 5 on both sides, the Raspberry Pi and the Arduino. The Raspberry starts by sending 0, which Arduino receives, prints and sends back adding 1, which is again sent back from Raspberry Pi as 11, and so on.

Figure 6: Number exchange code for Raspberry Pi in Rust.

The screenshot shows the Arduino IDE interface. On the left, the code for 'Task_5' is displayed:

```

Task_5
/*
  Giacomo Menchi
  IA 5
*/

#include <Wire.h>
int x;

void setup() {
  Wire.begin(0);
  Serial.begin(9600);
  Wire.onReceive(receiveEvent);
  Wire.onRequest(sendEvent);
}

void receiveEvent(int howMany) {
  while(Wire.available() > 1) {
    char c = Wire.read();
  }
  x = Wire.read();
  Serial.print(x);
}

void sendEvent() {
  Wire.write(<x>);
}

void loop() {
  delay(100);
}

```

On the right, a terminal window titled 'j/dev/cu.usbserial-1410' shows the output of the serial communication:

```

0
11
22
33
44
55
66
77
88
99

```

Below the terminal window are several status indicators: 'Autoscroll' (checked), 'Show timestamp' (unchecked), 'Newline', '9600 baud', and 'Clear output'.

Figure 7: Number exchange code for Arduino in C.

1.6 Task 6

Task: Set Serial communication between the RPi and Arduino using a logic level converter (LLC).

NOTE; check the book page 347 for UART and 359 page for LLC

Schematic: The picture below illustrates the task 6 schematic, where the Raspberry Pi is communicating with the Arduino via logic level converter and the Raspberry Pi is also powering the Arduino itself.

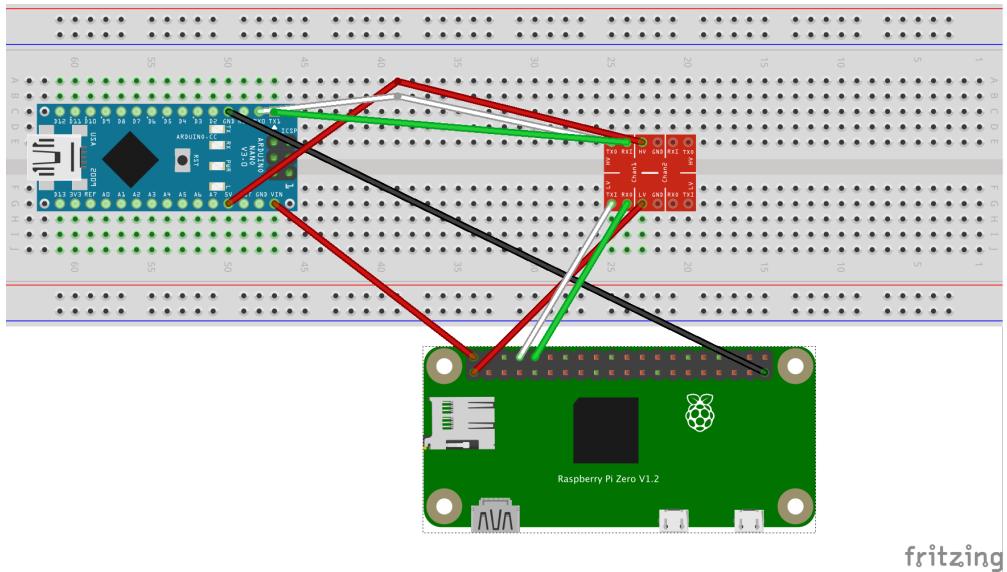


Figure 8: Raspberry Pi and Arduino Serial communication schematic.

1.7 Task 7

Task: Write a code for Serial connection on the Arduino side to send data to RPi.

Explanation: Said code can be found in the Github repo, inside the file named "Task_7_8.ino". The screenshot of it working can also be found below in task 8 (the two tasks were very similar, so I merged their screenshots).

1.8 Task 8

Task: Create some code in Rust for RPi to get data from Arduino over Serial. Send your name and kennitala to your Raspberry Pi.

Screenshot: As always, the screenshots below illustrate both the Raspberry Pi and Arduino side of the task 8.

```

Activities > Visual Studio Code
File Edit Selection View Go Run Terminal Help
OPEN EDITORS
  main.rs
src > main.rs
main.rs
You, a few seconds ago | Author (You)
1 use std::io;
2 use std::embedded_uart::uart::Uart;
3 use std::time::Duration;
4
5 fn main() {
6     let mut uart: Uart = Uart::new(baud_rate: 115_200, Parity::None, data_bits: 8, stop_bits: 1).expect(msg: "Error creating UART");
7     uart.set_read_mode(min_length: 1, timeout: Duration::default()).unwrap();
8     loop {
9         let read_string: String = uart.read_line().expect(msg: "Error reading bytes from UART");
10        println!("{}", read_string);
11    }
}
▶ Run | Debug
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
giacomenchi@ubuntu:~/Desktop/workspace/individual-assignment-5-JamesMaloney$ ./build_to_pi.sh
attempting to build program rust template and moving it to 192.168.1.131:/home/pi
Compiling rust template v0.1.0 (/home/giacomenchi/Desktop/workspace/individual-assignment-5-JamesMaloney)
Finished dev [unoptimized + debuginfo] target(s) in 0.38s
Building file: /home/giacomenchi/Desktop/workspace/individual-assignment-5-JamesMaloney/rust-template/target/debug/rust-template
Program uploaded, running...
-----
Name: Giacomo Menchi
Kennitala: 0507984029
~Connection to 192.168.1.131 closed.
giacomenchi@ubuntu:~/Desktop/workspace/individual-assignment-5-JamesMaloney$ 

```

Figure 9: Name and Kennitala exchange code for Raspberry Pi in Rust.

```

Arduino File Edit Sketch Tools Help
Task_7_8 | Arduino 1.8.13
Task_7_8
/*
 * Giacomo Menchi
 * IA 5
 */

void setup()
{
  Serial.begin(115200);
}

void loop() // run over and over
{
  Serial.print("Name: Giacomo Menchi");
  Serial.print("Kennitala: 0507984029");
  delay(2000);
}

Board at /dev/cu.usbserial-1430 is not available Copy error messages
Global variables use 230 bytes (1%) of dynamic memory, leaving 1818
Board at /dev/cu.usbserial-1430 is not available
16 Arduino Nano on /dev/cu.usbserial-1430

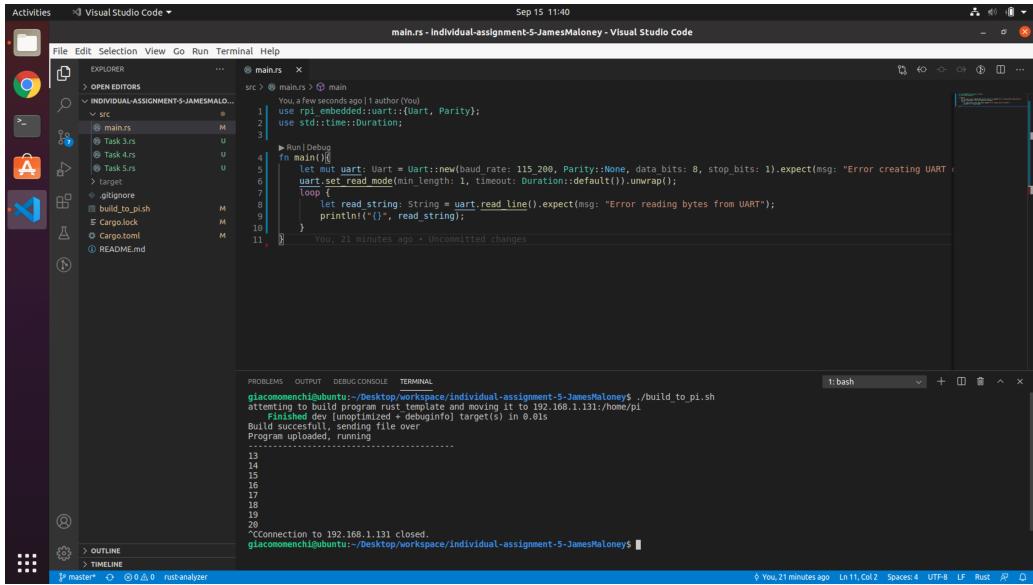
```

Figure 10: Name and Kennitala exchange code for Arduino in C.

1.9 Task 9

Task: Make Arduino send a string of numbers and RPi print the string. Let the new numbers increase with a new iteration.

Screenshot: No explanation required, just a numbers string sent from Arduino, printed by Raspberry Pi and sent again increased by one by Arduino each time.



The screenshot shows a Visual Studio Code interface with a dark theme. The left sidebar has icons for file, folder, search, and terminal. The main area shows an 'EXPLORER' view with a project named 'INDIVIDUAL-ASSIGNMENT-5-JAMESMALONEY'. Inside 'src', there are files: 'main.rs', 'Task 3.rs', 'Task 4.rs', 'Task 5.rs', 'Cargo.toml', and 'README.md'. The 'main.rs' file is open in the editor, showing the following Rust code:

```
You, a few seconds ago | (author: You)
1 | use rpi_embedded::uart;
2 | use std::time::Duration;
3 |
4 | fn main() {
5 |     let mut uart: Uart = Uart::new(baud_rate: 115_200, Parity::None, data_bits: 8, stop_bits: 1).expect(msg: "Error creating UART");
6 |     uart.set_read_mode(min_length: 1, timeout: Duration::default()).unwrap();
7 |     loop {
8 |         let read_string: String = uart.read_line().expect(msg: "Error reading bytes from UART");
9 |         println!("{}", read_string);
10 |     }
11 | }
```

The status bar at the bottom indicates 'You, 21 minutes ago * Uncommitted changes'. Below the editor is a 'TERMINAL' tab showing the command line output:

```
giacommenchi@ubuntu:~/Desktop/workspace/individual-assignment-5-JamesMaloney$ ./build_to_pi.sh
attempting to build program rust template and moving it to 192.168.1.131:/home/pi
Finished dev [unoptimized + debuginfo] target(s) in 0.01s
Build step 'Compile - sending file over' finished: Program uploaded: running.....
13
14
15
16
17
18
19
20
^CConnection to 192.168.1.131 closed.
giacommenchi@ubuntu:~/Desktop/workspace/individual-assignment-5-JamesMaloney$
```

The status bar at the bottom of the terminal tab shows 'You, 21 minutes ago Ln 11, Col 2 Spaces: 4 UTF-8 LF Rust'.

Figure 11: Increasing numbers string exchange code for Raspberry Pi in Rust.

1.10 Task 10

Task: Extract numbers from the string to integers and print them separately.

Screenshot: In this last task, I modified the previous code to make it print the single digits separately, as integers.

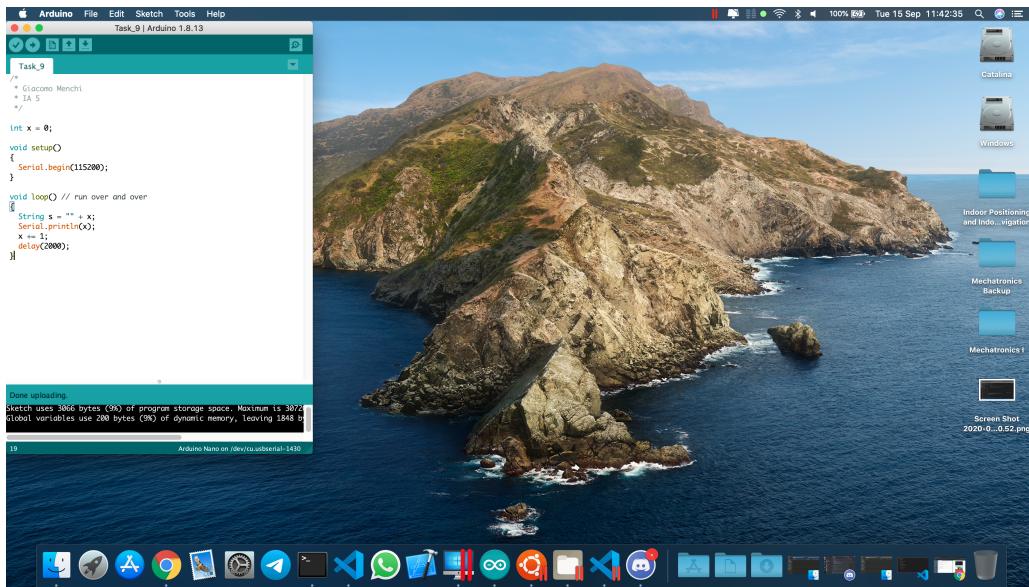


Figure 12: Increasing numbers string exchange code for Arduino in C.

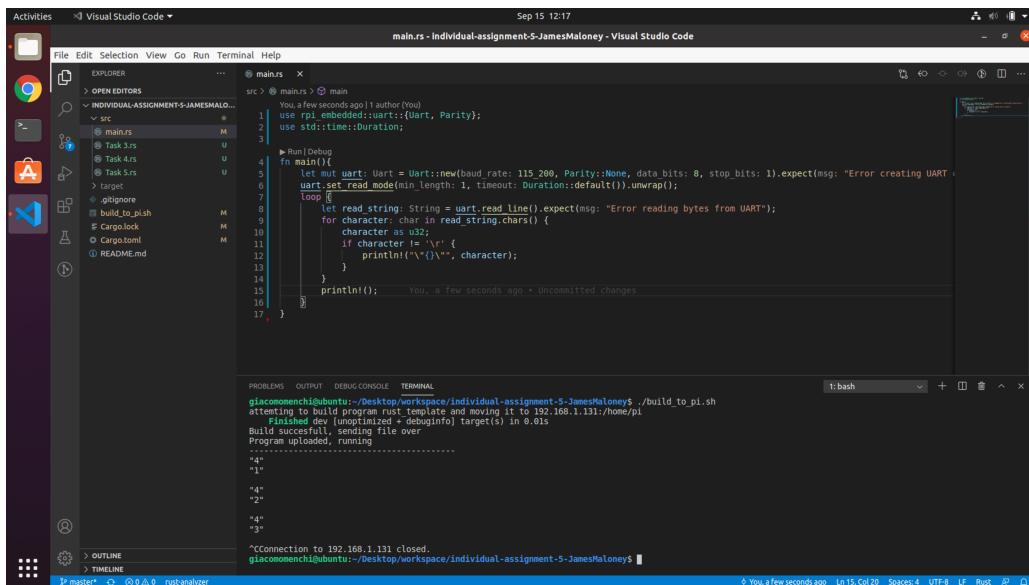


Figure 13: Printing numbers strings as single integers.