CHALMERS |  GÖTEBORGS UNIVERSITET

Development of Embedded and Real-Time systems

Work Package 5

Group 14

Eemil Jeskanen gusjesee@student.gu.se
Krasen Anatoliev Parvanov gusparkr@student.gu.se
Chrysostomos Tsagkidis gustsach@student.gu.se

Program:	Software Engineering and Management, BSc
Course:	DIT632: Development of Embedded and Real-Time systems
Date:	2021/02/25
Number of pages:	11

Table of Contents

Exercise 5 part 1

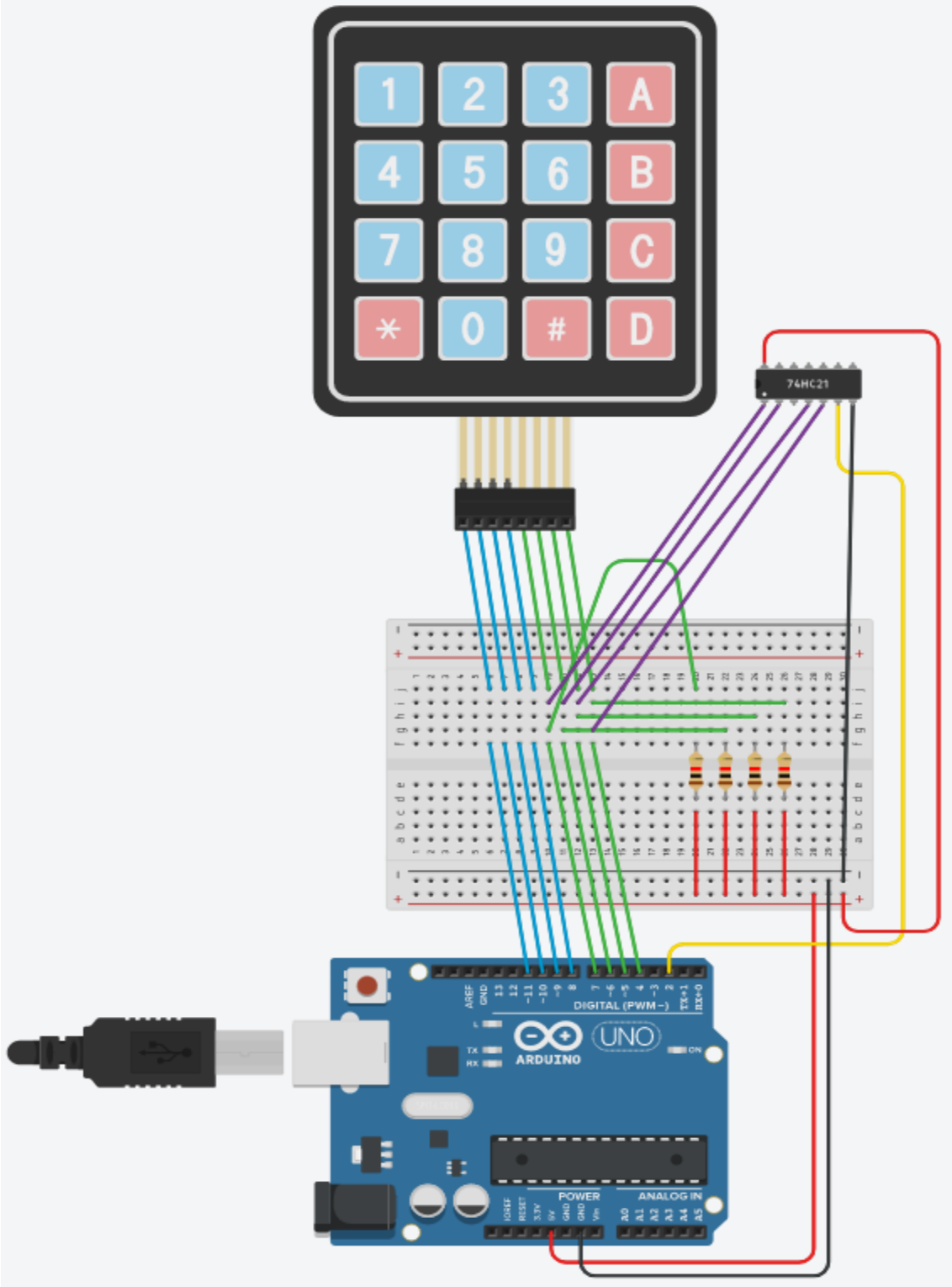
Code

```
/* ====================================
File name: exerc_5_1.ino
Date: 2021-XX-XX TODO edit
Group nr 14
Members that contribute to the solutions: Krasen Parvanov,
Chrysostomos Tsagkidis, Eemil Jeskanen
Member not present at demonstration time:
Demonstration code: TODO edit
==================================== */

/* --- Macros predefined for the compiler
DDRB  Data direction register B
PORTB Outport B
PINB  Inport B
DDRD  Data direction register D
PORTD Outport D
PIND  Inport D
HIGH ON-1
LOW OFF-0
*/

// Define section
#define NUMBER_OF_ROWS 4 // number of rows in the keypad
#define NUMBER_OF_COLUMNS 4 // number of columns in the keypad
#define DELAY_AFTER_PRESS 500 // default delay after pressing a button
#define ENABLE_OUTPUT_REGISTER 0b00001111 // use to set the DDRB
register as output
#define SET_INPUT_REGISTER 0b00000000 // use to set the DDRD register
as input
#define NOT_PRESSED 0 // used for checking if a key has not been
pressed
#define BAUD_RATE 9600 // define baud rate for serial

/* ================================== Main program section
==================================== */
/* This program is designed using Tinkercad(image of the circuit can
be seen image bellow) simulation for Arduino Uno.
```

```
* The programme handles reading the keypad buttons the program prints out the pressed key number in the serial monitor and starts a 1s delay,
* if no key is pressed nothing is printed out. The keypad is designed as a 4-4 Matrix with mapping the buttons as follows 0-9->A-F.
* The programme uses interrupt instead of polling in order to decrease the usage of processing power in the main loop.
* */

// Define constants
const unsigned char keypad[NUMBER_OF_ROWS][NUMBER_OF_COLUMNS] = { // keypad matrix to map the buttons in order 0-9->A-F
       {'0','1','2','3'},
       {'4','5','6','7'},
       {'8','9','A','B'},
       {'C','D','E','F'}
};
const unsigned int columns[NUMBER_OF_COLUMNS] = {7, 6, 5, 4}; // define the column's pins numbers
const unsigned int rows[NUMBER_OF_ROWS] = {11, 10, 9, 8}; // define the row's pins numbers
const int interruptPin = 2; // define the interrupt pin to PIN2 of PORTD

// Global variables
volatile unsigned char input; // store the pressed button value
volatile unsigned long lastInterrupt; // Use to keep track of the time of the last occurred interrupt

// Set-up section
void setup()
{
   Serial.begin(BAUD_RATE); // start and configure the serial monitor with baud rate
   DDRD = SET_INPUT_REGISTER; // setting the data direction register for port D to input
   DDRB = ENABLE_OUTPUT_REGISTER; // setting the last for bits(pins) of data direction register for port B to output(1s)
   lastInterrupt = 0; // initialize the last interrupt to default value of 0
   attachInterrupt(digitalPinToInterrupt(interruptPin), checkForKeypadInput, FALLING); // attach interrupt to the interruptPin and define the interrupt trigger as Falling(LOW)

// and attach the method to check for keypad button on the interrupt
```

```
}

// Main Loop
void loop()
{
   if(input != NOT_PRESSED){ // check if button has not been pressed
       Serial.println((char) input); // if pressed print the pressed
character
       input = NOT_PRESSED; // reset the input to NOT_PRESSED
   }
   PORTB = SET_INPUT_REGISTER; // reset the input register
}
// Function to check which keypad button is pressed and return it if
any
void checkForKeypadInput()
{
   if (millis() - lastInterrupt > 10) { // check if 10ms have passed
since the last interrupt to avoid multiple iteration based on hardware
behavior

       input = NOT_PRESSED; // used to store the found charectred set
by default to Not-pressed
       unsigned int i,j; // indexes for the loop iterations

       for(i = 0; i < NUMBER_OF_ROWS; i++){ // start a loop to iterate
over the rows
           for(j = 0; j < NUMBER_OF_COLUMNS; j++){ // start iterating
over the columns for the current row
               if(digitalRead(columns[j]) == LOW &&
digitalRead(rows[i]) == LOW){ // check if the column is LOW, when LOW
the button that connects row and column is pressed (because of the
PULL-UP resistor)
                   input = keypad[i][j]; // key is found and save the
value from the matrix in the variable

               }
           }
           digitalWrite(rows[i], HIGH); // set back the current row to
HIGH (activate)
       }

       lastInterrupt = millis(); // update the lastInterrupt time
stamp
   }
}
```

Circuit image



Exercise 5 part 2

Code

```
/* ===================================
File name: exerc_5_2.ino
Date: TODO
Group nr 14
Members that contribute to the solutions: Krasen Parvanov,
Chrysostomos Tsagkidis, Eemil Jeskanen
Member not present at demonstration time:
Demonstration code: TODO
=================================== */

/* --- Macros predefined for the compiler
DDRB  Data direction register B
PORTB Outport B
PINB  Inport B
DDRD  Data direction register D
PORTD Outport D
PIND  Inport D
HIGH ON-1
LOW OFF-0
*/

// Define section
#define NUMBER_OF_ROWS 4 // number of rows in the keypad
#define NUMBER_OF_COLUMNS 4 // number of columns in the keypad
#define DELAY_AFTER_PRESS 500 // default delay after pressing a button
#define ENABLE_OUTPUT_REGISTER 0b00001111 // use to set the DDRB
register as output
#define SET_INPUT_REGISTER 0b00000000 // use to set the DDRD register
as input
#define NOT_PRESSED 0 // used for checking if a key has not been
pressed
#define BAUD_RATE 9600 // define baud rate for serial
#define VOLTAGE_OFFSET 500 // voltage offset for TMP36 in millivolts
#define BUTTON_MESSAGE "Button pressed: " // string to print
#define TEMPERATURE_SCALE_MESSAGE " Celsius" // used to display the
temperature type
#define TEMPERATURE_MESSAGE "Temperature is: " // used for displaying
before the temperature
```

```
#define VOLTAGE 5000 // arduino power voltage
#define ANALOG_UNITS 1024.0 // used to calculate milli voltage for
arduino analog signal

/* ================================== Main program section
================================== */
/* This program is designed using Tinkercad(image of the circuit can
be seen bellow) simulation for Arduino Uno.
* The programme handles reading the keypad buttons the program prints
out the pressed key number in the serial monitor and starts a 1s
delay,
* if no key is pressed nothing is printed out. The keypad is designed
as a 4-4 Matrix with mapping the buttons as follows 0-9->A-F.
* In addition to that the programme uses a TMP36 sensor to read the
temperature. The programme displays the temperature in Celsius once
it's starts,
* after that every time the keypad is pressed the programme reads the
thermo-sensor again and recalculates and displays the temperature.
* */

// Define constants
const unsigned char keypad[NUMBER_OF_ROWS][NUMBER_OF_COLUMNS] = { //
keypad matrix to map the buttons in order 0-9->A-F
       {'0','1','2','3'},
       {'4','5','6','7'},
       {'8','9','A','B'},
       {'C','D','E','F'}
};
const unsigned int columns[NUMBER_OF_COLUMNS] = {7, 6, 5, 4}; //
define the column's pins numbers
const unsigned int rows[NUMBER_OF_ROWS] = {11, 10, 9, 8}; // define
the row's pins numbers
const int analogPin = A0;  // define input analog pin on PORTA
const int interruptPin = 2; // define the interrupt pin to PIN2 of
PORTD

// Global variables
volatile unsigned char input; // store the pressed button value
volatile unsigned long lastInterrupt; // Use to keep track of the time
of the last occurred interrupt
// Set-up section
void setup()
{
   Serial.begin(BAUD_RATE); // start and configure the serial monitor
with baud rate
```

```
   DDRD = SET_INPUT_REGISTER; // setting the data direction register
for port D to input
   DDRB = ENABLE_OUTPUT_REGISTER; // setting the last for bits(pins)
of data direction register for port B to output(1s)
   lastInterrupt = 0; // initialize the last interrupt to default
value of 0
   attachInterrupt(digitalPinToInterrupt(interruptPin),
checkForKeypadInput, FALLING); // attach interrupt to the interruptPin
and define the interrupt trigger as Falling(LOW)

// and attach the method to check for keypad button on the interrupt
   readAndDisplayTemperature(); // call method to read the temperature
and display it
}
// Main Loop
void loop()
{
   if(input != NOT_PRESSED){ // check if button has not been pressed
       Serial.print(BUTTON_MESSAGE); // print button start message
       Serial.println((char) input); // if pressed print the pressed
character
       input = NOT_PRESSED; // reset the input to NOT_PRESSED
       readAndDisplayTemperature(); // call method to read the
temperature and display it
       delay(DELAY_AFTER_PRESS); // delay
   }
   PORTB = SET_INPUT_REGISTER; // reset the input register
}
// Function to read the analog signal from the sensor and display the
corresponding temperature value in Celsius
void readAndDisplayTemperature(void) {
   int readValue = analogRead(analogPin); // read the thermo-sensor
and save the value
   float celsius = calculateTemperatureInCelsius(readValue); // call
function to calculate the the temperature value in Celsius and save it
   Serial.print(TEMPERATURE_MESSAGE); // display the temperature
message
   Serial.print(round(celsius)); // display the temperature rounded to
the closest number
   Serial.println(TEMPERATURE_SCALE_MESSAGE); // display the degree
scale

}
```

```
// Function to check which keypad button is pressed and return it if
any
void checkForKeypadInput()
{
   if (millis() - lastInterrupt > 10) { // check if 10ms have passed
since the last interrupt to avoid multiple iteration based on hardware
behavior

       input = NOT_PRESSED; // used to store the found charectred set
by default to Not-pressed
       unsigned int i,j; // indexes for the loop iterations

       for(i = 0; i < NUMBER_OF_ROWS; i++){ // start a loop to iterate
over the rows
           for(j = 0; j < NUMBER_OF_COLUMNS; j++){ // start iterating
over the columns for the current row
               if(digitalRead(columns[j]) == LOW &&
digitalRead(rows[i]) == LOW){ // check if the column is LOW, when LOW
the button that connects row and column is pressed (because of the
PULL-UP resistor)
                   input = keypad[i][j]; // key is found and save the
value from the matrix in the variable
               }
           }
           digitalWrite(rows[i], HIGH); // set back the current row to
HIGH (activate)
       }

       lastInterrupt = millis(); // update the lastInterrupt time
stamp
   }
}
// This function calculates the temperature in Celsius given a analog
reading
float calculateTemperatureInCelsius(int inputVoltageReading) {

   // Convert the analog reading (which goes from 0 - 1023) to a
voltage (0 - 5000mV):
   float milliVoltage = (inputVoltageReading * (VOLTAGE /
ANALOG_UNITS));
   float temperature = (milliVoltage - VOLTAGE_OFFSET)/10;  //
calculate the temperature by deducting the offset and dividing by 10
mv per degree
   return temperature; // return the temperature value in Celsius
}
```

Circuit Image