Exercises and assignments for the course DIT632 Development of Embedded systems
VT20 _ Part 2 of 3

Exercise points: For each of the handed in and approved exercises you get exercise points from zero up to a maximum specified point. You can get a total sum of 51 exercise points for all handed in exercises during the course. Conditions to pass the sub course "Assignments" and to get bonus points to final written exam is described in part 1 of Exercises and assignments.

Terms of assignments and bonus
All exercises that are submitted (possible for all in WP2 - WP6) must meet the following requirements to give bonus points at final exam.
Includes a program header as below:
/ * ==========
File name: exerc_x_y.c (or cpp)
Date: 2020-mm-dd
Group Number:
Members That contributed:
xxxxxxx xxxxxxxx
yyyyyy YYYYYYYYYY
zzzzz ZZZZZZZ
Members not present at the demonstration:
Xxxxxxx
Demonstration code: [<Ass code xxxxx>] **Important , No code no exercise points !**
========== * /

All submissions must be handed in by one of the group members via course homepage. Before handed in all programs should be demonstrated for a teaching assistant (TA) and if the TA approves the program you will get a specific unique demonstration code <xxxxx> to type into the program information header.
You should only hand in one file per WP nr. All programs should be included in one xxx.zip file. If there are any solutions containing more than two files you had to put them in a separate directory.

Work package nr 4/ Low level C-programming (Max 11p)

Introduction information for this course week

For this week tasks we will, for some exercises, use a specific web based IDE environment, **Tinkercad** supporting virtual designing of electrical circuits built up by discreet components (resistors, LED:s , keyboards …) and even a CPU board (Arduino). The entire design can then be tested by simulating the systems function in an associated simulator. You can by this easily test your own software solutions for a specific hardware design.

A more comprehensive description of Tinkercad can be found later in this document. First there is some more general C programming exercises.

Exerc_4_ 1 **(**Filename code.c) (2p)

Pack and unpack variables into a byte. You need to store 4 different values in a byte. The values are:

Name	**Range**	**Bits**	**Info**
engine_on	0..1	1	Is engine on or off . This is bit no 7 (MSB)
gear_pos	0..4	3	What gear position do we have
key_pos	0..2	2	What position is the key in
brake1	0..1	1	Are we breaking (told by sensor 1)
brake2	0..1	1	Are we breaking (told by sensor 2) = bit no 0 (LSB)

We should store them in a byte like this:

[engine_on]	[gear_pos]	[key_pos]	[brake1]	[brake2]	
1 bit	3 bits	2 bits	1 bit	1 bit	(LSB, Least significant bit)

(8 bits in total)

Write a program ***code.c*** that takes 5 arguments (less or more should be treated as an error). The arguments should correspond to the values/variables above.
Example for a start of the program from command line:
code 1 2 2 1 1 (In the console window for Windows, ./code 1.. in Linux)

The above should be treated as:

Name	Value	
engine_on	1	Bit no 7
gear_pos	2	
key_pos	2	
brake1	1	
brake2	1	Bit no 0

The program should pack these values together in a byte (unsigned char) and print it out to the console in hexadecimal form. For this example, it should be 'AB' corresponding to bits '10101011. After this your program should exit. If your program finds anything wrong (too many/few arguments, faulty input values..) your program should print out an error message and exit.

Exerc_4_ 2 (decode.c) (2p)

Write a program ***decode.c*** that takes 1 argument , in the example 'AB' (less or more argument should be treated as an error). The arguments should correspond to the byte as printed by your earlier code program. If your program finds anything wrong (too many/few arguments, faulty input values..) your program should print out an error and exit.

The program should unpack the bytes according to the specification above. Print out the result as below:

Start program in the console window : ***decode AB*** *(If* Compiled to code.exe)
Should print out.

Name	Value
engine_on	1
gear_pos	2
key_pos	2
brake1	1
brake2	1

Exerc_4_3 (File name exerc_4_3.c) (2p)

In this exercise you shall design a general C-program that intends to read a smart keyboard including some internal 8 bits registers. The keyboard is connected to the CPU and all keyboards registers can be accessed by reading or writing to the register specific address.

The program can read the value of a pressed key on the keyboard as a 8-bit value in a register with a specific address. The value (0-15) of a pressed key is the value of bit 0 – 3 in the register. If a key is down when the register is read bit 7 (DAV in figure) is zero otherwise the bit 7 is set (1).

In figure the key wit value 6 is down for the moment.

If no key pressed when reading the register DAV bit is set (1).

	DAV				Key nr			
	0	x	x	x	0	1	1	0
Bit nr	7	6	5	4	3	2	1	0

To simulate this reading of the keyboard without any hardware for a general C-program we will use some C-program help-functions. You can find these at the course homepage in Files. / Program examples… ../Course week #4.

In the file : bit_manage_func.c you can find the following functions**:**

void f_delay(int tenth_sec) : Gives a time delay for a number of tenth of seconds.

unsigned char random_inport(void) : Generates a random value 0 – 255 .

This will simulate the read value from a 8 bit inport in a system.

void printport(int portvalue) : Prints out the binary pattern and the decimal value for a char of value 0 – 255. (Not used in this exercise).

The keyboard is controlled by a hardware and to read a pressed key number you read the 8-bit value in a register at a specific address. In our case we simulate this by call the function ***random_inport()***. The value in register should be interpreted as follows: If bit no 7 is zero there is a key pressed and the key nr (0 – 15) can be read as bit 0 to 3 in the value. The value on bit no 4 to 6 is of no interest and its value should not affect the program function.

Write a program that periodical (once every half second) reads the keyboard register (the random number). If a key is pressed (bit-7) read the key nr and print it out on the console in hexadecimal form 0 – 9, A- F. If no key pressed there shouldn't be any printout.

Introduction to Arduino, ATmega microcontroller and TinkerCad

To test the solution of the exercises exerc_4.4 and 4.5 and later some more we will use a web environment Tinkercad and a hardware build around an Arduino Uno card. Before you start solving the exercises you need some information about things around the environment, hardware and how we should design the hard- and software.

Arduino CPU board

Arduino is a small computer board designed to make it easy and quickly to develop small embedded systems. The various Arduino boards are designed with different ATmega microcontrollers. Arduino Uno, which we will use for some exercises in this course, is equipped with an ATmega 328 microcontroller.

The Arduino system also includes its own development environment (IDE) which, with help of extensive associated library functions, facilitates software development and makes it easy for a beginner to develop the software for a hardware system in the program language C supplemented by all Arduino's own library functions and without knowing much about lots of low-level details.

The Arduino IDE is a freeware and the only thing you need to bay is an Arduino CPU board (from 250 skr) and the additional electronic components you need for your specific own hardware system.

In this course we will not use the real Arduino CPU board, but we will build electronic circuits including Arduino Uno CPU board in a virtual environment supported by Tinkercad.

You may need some information of developing C-programs for an Arduino system in the following exercises. You find all this information on the Arduino homepage (https://www.arduino.cc/).

Tinkercad

In some of the exercises in WP # 4 and 5 we will use the development environment, **Tinkercad**, which makes it possible to virtually build an electronic system around an Arduino card computer and then write the software for the system. The entire design can then be tested by simulating the systems function in an associated simulator. You can easily test your own software solutions for a specific hardware design.

Tinkercad is a web-based development environment which makes it independent of the user's own computer environment. To use Tinkercad you only need to create an account on their website. You can if you want create one common account for your work group.

Link to Tinkercad: https://www.tinkercad.com/

Under the menu **Circuits** you will find the development environment that we will use in the exercises. In this IDE you can design many electric circuits with help of a selection of electronic components and an Arduino CPU board. When choosing the components, you should choose **all in menu** to find the needed components.

To design the circuits in the virtual environment you need to use a virtual **breadboard** that you also find in the component list.

If you never used a breadboard se the first 5 to 6 minutes of this video:

https://www.youtube.com/watch?v=oiqNaSPTI7w

To start develop a circuit.

Log in to your account in Tinkercad. You can study a lot of electronic circuits in the Gallery/Circuits and you can copy them to your own account to study and test.

From Tinkercad home page, chose Circuits in left menu and then Create new. In right menu Components chose view all. From the list you can chose any listed component, breadboard, Arduino CPU bord and then start connecting the components with wires. Normally it´s very easy to connect with wires but it can sometime look like a connection but it isn´t connect to each other. To test you can try to move the component and if the wire follows the component it´s ok.

Briefly about the ATmega 328 microcontroller

First some general about micro controllers. A Micro controller is a bit different designed in comparison to a more general processor. They use to have ready-made IO ports for both digital and analog signals. They also often have built-in modules for AD/DA converters, timer circuits, etc. The different IO ports have fixed addresses. IO-ports and the internal modules are easily configured by writing bit patterns in special registers. ATmega 328 has many digital 8 bits IO ports (On Arduino one 8 (PORTD) and one 6 bits digital IO PORTB) as well as many analog inputs and outputs with associated AD and DA converters, respectively.

The digital IO ports are configured for In or Out on bit level by writing 1 or 0 in the corresponding bit in a so-called **data direction register DDRD and DDRB** respectively. Writing and reading can be done if you know the address of the ports. In Arduino's IDE you can, via simple functions call or use of so-called Macron, configure the ports to the desired function and then easily read and write on the IO ports.

You can find a pin-layout picture fort the use of ATmega328 on the Arduino Uno computer board at:

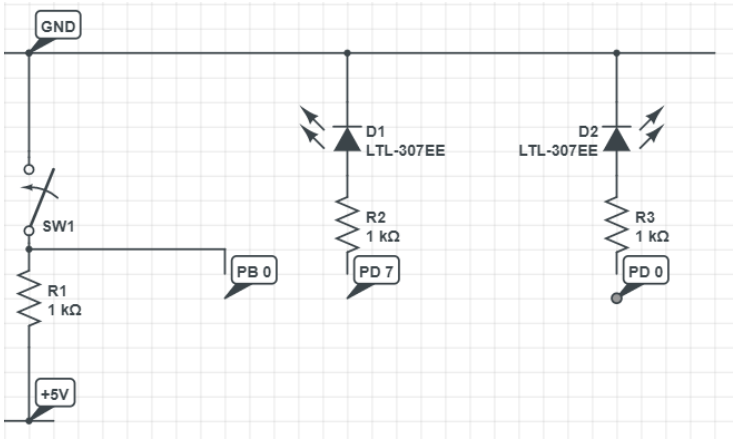
https://i2.wp.com/marcusjenkins.com/wp-content/uploads/2014/06/ARDUINO_V2.png

On the course homepage in the file archive you can find some documents regarding Arduino and the ATmega microcontroller. One pdf describes the controller and it´s memory and register structure.

Exerc_4_4 **(** File name exerc_4_4.c**)** **(2p)**

In Thinkercad you should drew a new circuit as the figure on next page views. In the circuit PD0-PD7 is used as out bits and connected to the LED: s anode side via a resistor (1kOhm). Bit PB0 is used as an in-bit reading the status from one side of a DIP switch. The value to PB0 will be either 1 or 0 depending on if the DIP switch is on or not.
The circuits principle is as figure to right. A high (1 (5V)) value on an out-bit on PD will light up the diode.

When drawing the circuit, the code window should be closed.



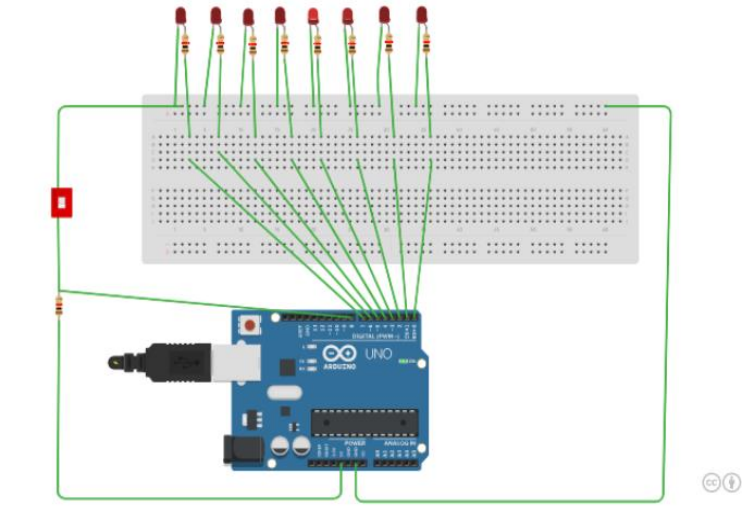
Write then a program with the function described below that uses out- and in-port as described above. Code window should be in text mode.

The program should work as follows:

1. Configure the PORTB to inport and PORTD as out port.
2. Write 3 (binary 0000 0011) on the OUT PORTD.
3. In an infinitive loop:
   Read bit PB0 on the IN_PORTB
   - If bit is set (1) rotate the content on OUT-port one step left. If bit 7 on OUT PORTD is set before the left shift bit 0 should be set after the shift. (we rotate the byte content)
   - If bit is clear (0). Do not change the value on OUT-port.

For the delay use the delay function as shown in the program skeleton below.

You should use the predefined macros for reading/writing to the registers and in/out ports. For some reason you should use PORTx when writing out values and PINx for reading the same port if configured as inport.



It is possible to use the general way of accessing the registers and ports but as they are both in and out opportunities for the same port it is a bit tricky to get it work well.

With the named macros it usually works in a simple way to read all bits in a port.

For some other reasons we shall use the predefined structure with the two functions setup() and loop() .

Run the program in the simulator to test the function.

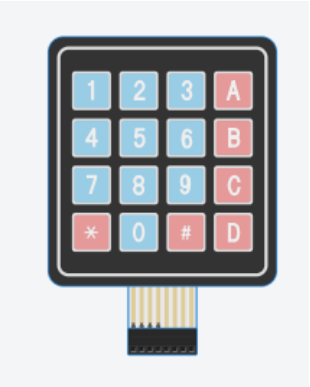
Note : If you get some problems it can help using the debugger. Before start set a break point. There is also a possible to connect a Multimeter to any part of the circuit. Depending of our use of bit PD1 and PD0 we can´t use the serial monitor for printing.

When ok you just copy the code to any editor and add necessary information in the program head, save it and hand it in as usually together with the other solutions.

```
// ---- Program template for Arduino in Tinkercad VT 2020
#include <avr/io.h>
#include <util/delay.h>

/* --- 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
*/
unsigned char input;
void setup() {
  Serial.begin(9600); If use of Serial monitor
  DDRD = DDRD | B00000010;  // Sets pin 2-7 as output-bits 0 . Bit 0 and 1 in this way if use of Serial monitor.
  DDRB=  …….

}
```

```
void loop() {
  // ------ Main loop-------


  delay(500); // Wait for 500 millisecond(s)


}
```

Exerc_4.5 (Filename: exerc_4_5.c): Keyboard scanning. (3p)

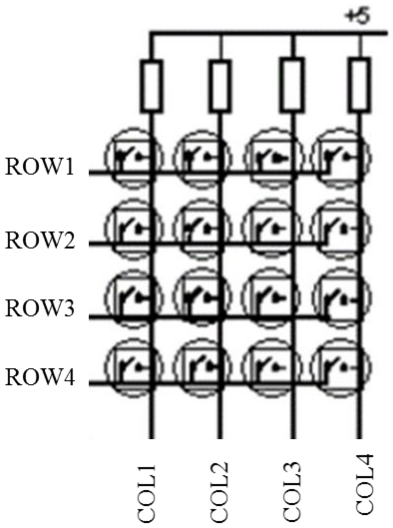


Some cheap keyboards are not connected to a supporting hardware with register for reading the key number. This simpler device is a bit harder to read. In TinkerCad you can connect such a simple keyboard (Figure) to Arduino Uno and then design a program that reds a pressed key number.

The principle for the keyboard and reading a key.

The keyboard is built up by four row connectors and four column connectors. At the 16 crosses between raw and column there is normally no electrical connection. At each cross there is a key and if a key is pressed there will be an electrical connection between the column and raw. (See figure below)

The rows (row 1 to row 4) is possible to connect to anything via the four left bits in the flat connector. On the four right bits the four columns are possible to connect to anything.



The keyboard can be connected to any of Arduino´s IO-ports. One possibility is to connect all 8 row/columns to Port D and configure the port D as 4 bit out/four bit in.

If we chose this way of connecting the keyboard it will be a bit more tricky than necessary to develop the program for reading the keyboard. This is depending of the possibility to use the same physical IO-port as IN or OUT on bit level. We will also lose the possibility to use the serial monitor for printing out some control text.

Instead we chose to connect the keyboard by connecting the rows to Arduino Port B (bit 0 – 3) and the Columns to Port D (bit 0 – 3) . By this we can use Port B as outport and Port D as inport.

To ensure the level (electric) of the columns, when no key is pressed, you normally connect them to 5V via a resistor (so called “pull up resistor). You can see this in the figure above.

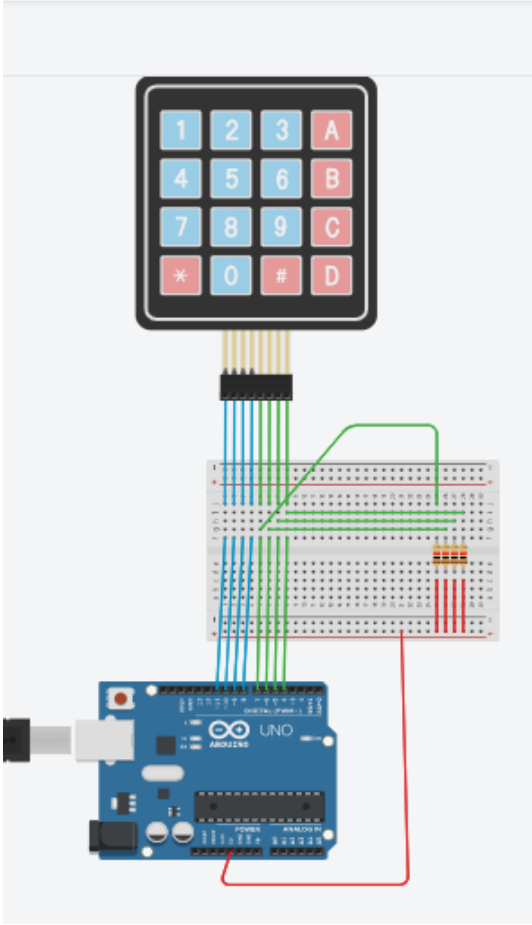
One way to detect a pressed key (if pull up resistors) is to set a row to low (0) and the other row bits to high (1) via port B. If any key in that low row now is pressed it will give the corresponding column bit a clear value (0) and the other bits of port D high values. This can be detected by reading the column inbits (PD3-PD0). If no key in the specific row is pressed the value from the column-bits will be high on all four bits.

To read the keyboard you must in a loop set one row to low and the others to high and check if any column – bit is not equal to high. If still four bits high set next row to 0 and check the columns value. As soon as you find a column value with not four set bits you have detect a pressed key. If so, you know which row and column it is, and you can from that calculate which key number is pressed. The pattern for these two nibbles (nibble = four bits) is unique for each key. If you gone through all four rows without finding any column INPORT value not equal to four set bits there is no key pressed for the moment.

We can assume in this exercise that the Key is numbered from top row (row 1) from left 0, 1, 2, 3, 4 and next row 5, 6, 7,8 etc. The keys from 10 and up should be named in hexadecimal code as A, B ..

To do :

In Tinkercad you should Build a new circuit as described above and viewed in figure to right. We will later expand the circuit so it´s good to design it as the figure shows.



You should then, also in Tinkercad, develop a program reading the keyboard system above. The program should print out the key number in the serial monitor if a key is pressed. If no key is pressed there should be nothing printed out.
The program structure should be as below. You should also use the predefined macros as in earlier exercises. You can use the function Serial.println() for printing on the monitor.

- configure the IO ports
- in an infinity loop
  Call a function checking if any key is pressed and if it should return it´s value.
  If a key was pressed print out the key value (0-9 , A-F) on the serial monitor.
  Delay for one second.

Demonstrate for TA and hand in the solution as for exercise 4_4 .

Work package nr 5/ Keyboard interrupt and simple temperature meter (6 p)

Exerc_5_ 1 : (3p + 1p)

In this exercise we will further develop the keyboard system from exercise 4_5.
The main idea is to, instead of continuously checking the keyboard, read the key in an interrupt service routine (**ISR**). The key reading function from earlier exercise should now be called as an interrupt service routine as soon as any key is pressed. The number of the pressed key should then be written out in the monitor by the main program.

You shall use the hardware interrupt INT0 on the Arduino system. To get an interrupt call you should use the pin named PD2 (INT0) on the Arduino board as input for the hardware interrupt.
To configure the system for manage of an interrupt you shall use the Arduino setup() function call :

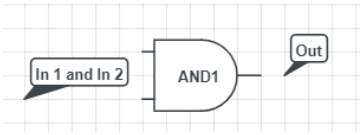
attachInterrupt (digitalPinToInterrupt(2), irqserviceroutin name , FALLING);

By this the function (a void type function) will be called if the level on the INT0 pin change from high to low level. (If named RISING instead of FALLING it will be called for the opposite level change).

To get a hardware indication of a pressed key we will use four input NAND gates, one for each column.

An AND gate (Logical And) is a digital circuit that realizes the AND logical function

The function can be described by a truth table showing all possible states for in and out as the table besides. The schematic symbols for an AND gate is as figure besides.



In 1	In 2	Out
0	0	0
0	1	0
1	0	0
1	1	1

A four input AND gate gives in the same way only high(1) out when all in are high (1, 1, 1, 1). All other combinations for in-signal will give a low out.
If we connect all four inputs in a AND- gate to the column wires of our keyboard system, the out-pin will be low for all combinations except from the state with four high input. This means that when no key is pressed or when a key in a row with high state is pressed the output from the AND gate will be high. The output will only be low when a key in a row with low state is pressed. If we connect the out-signal to the Arduino INT0 pin this will generate an interrupt request to the system.
A dual four input digital logic AND gate in a chip is normally labeled 74HC21. You can find such a chip in Tinkercad and connect it to our system. You find all input and output to each gate on the chip when pointing to them.

To do :

Connect a four input AND gate to your keyboard system as described above so a pressed key in a row with low value (0) will generate an interrupt.

To make the system generate an interrupt for any key pressed you will in the main program set all rows to low. Any key pressed will by that generate an interrupt and call the interrupt service routine (IRS). The IRS should then decode what key is pressed and “return” the key number to the main

program that print it to the serial monitor. If no key is pressed the main program just loop around as below pseudo code shows.

```
Setup(){
…..
}

Loop(){
  ….
 If(keyhit){
    Serial.println(keynumber);
    …..
}

void keyboardirq(void){
  // Read key number ------
  …..

}
```

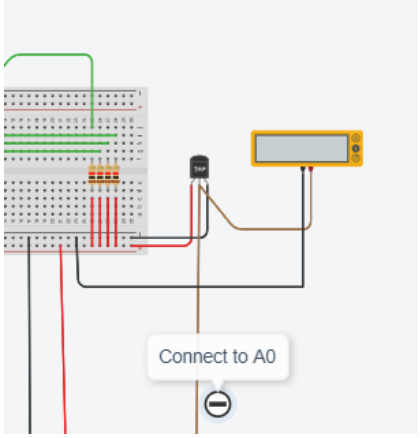
Examination of the solution:
Demonstrate the function for a TA and get the examination code. Put the code in the program header, copy the program list to a word document. Take a "picture" of the circuit and copy it to the same document.
Describe as god as possible what the below line is done:
attachInterrupt (digitalPinToInterrupt(2), irqserviceroutin name , FALLING);

Save the document as a pdf and hand in as a solution for exercise 5.

Exerc_5_ 2 : (Filename: exerc_5_2.c): Temperature meter (2 p)

In this exercise you should extend the function of your keyboard system with a temperature meter. To do so add a temperature sensor (TMP36) to the system as the figure shows.
The main idea is that the main-loop in the program, with short delay (500ms), check the temperature by reading the input voltage on AD – input A0. Calculates the corresponding temperature and print it out on the serial monitor. To calculate the temperature you have the following input data:

AD converter with 10 bits resolution. You can use the Arduino IDE:s function (analogRead()) to read A0.
Temp sensor TMP36: Output 750 mV at 25 °C, sensor gain 10 mV/ °C



You can also add the Multimeter to be able to check the printed temperature for a certain in-voltage**.**

Extend the main program from 5_1 with function so that the system normally writes out the temperature and if any press the keyboard the system reads the key and print out the key number. The out printing can be configured as you decide.

Note : Any solution haven´t yet been tested for this exercise. If you notice any problems with your solution try to analyze the cause of the problem

Examination : Demonstrate for a TA and write the examination code in the program head. Copy the program to the same document as exercise nr 5_1

Work package nr 6 : (Total points (prel) 12 p)

Wil be delivered in a separate pdf DIT632 Exercises_P3_v20xxxx.pdf later on .