# Real Time Systems and Programming Languages

# AIE2/EN4

## Prototype of a Coffee Vending Machine

## Final MiniProject

**Instructions**: The project **should be done in groups**. You need to write a final miniproject report with a maximum of 5 pages **explaining your program design decisions**. **The source code of your miniproject should be attached in a separate file.** An extra front page (not included in the 5 pages count) should be included, containing the name of the persons in your group and which programming task(s) each member of the group has performed in the miniproject. **Your miniproject report should be uploaded into digital exam before the deadline.**

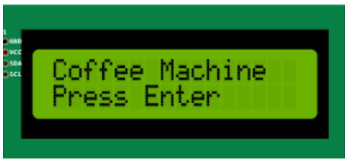
**Requirement: Your program should fulfil the specifications that are stated in the last page of this document and the code should be of your own.**

**I strongly suggest that you to test your design in the simulator first and then build and test it with real hardware. In the real hardware implementation you may choose a different type of display, any other display that is available in the lab will work but in the simulation use the LCD with I2C communication.**

**Description:**

Design and build the prototype of a simple coffee vending machine (SCVM) with a user interface in HW and an open loop controller of a servo motor that will move with different degrees to simulate serving some type of coffee.

Your program should show on a LCD in the beginning the following:

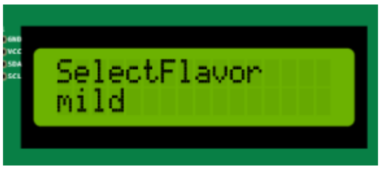


Coffee Machine
Press Enter

User should press a push button called enter to continue. Assume that your coffee machine offers the following types of coffee:

MILD, BOLD, STRONG, FULL

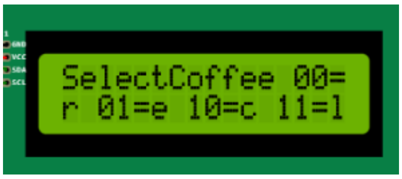
Your SCVM should have a sliding potentiometer to allow you select the type of coffee roast, as the following figure shows (in the example is mild):



SelectFlavor
mild

You will need to convert the voltage from the potentiometer to one of the 4 values MILD, BOLD, STRONG, FULL.

After the user presses again the enter push button, your LCD should allow you to select the type of product



```
SelectCoffee 00=
r 01=e 10=c 11=l
```

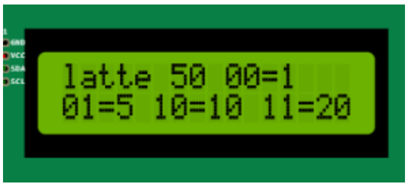
The user will be able to select the type of coffee with 2 **coffee selection switches** with following code

| **Coffee type** | **Coffee Selection Code** | |
|---|---|---|
| Regular (**r**) | 0 | 0 |
| Espresso (**e**) | 0 | 1 |
| Cappuccino (**c**) | 1 | 0 |
| Latte (**l**) | 1 | 1 |

These products will have the following prices:

| **Coffee Type** | **Price** |
|---|---|
| Regular | 16 DKK |
| Espresso | 19 DKK |
| Cappuccino | 25 DKK |
| Latte | 50 DKK |

After selecting the product, the used should pay for his selection as show bellow. In this example for a selected latte the price is 50 DKK

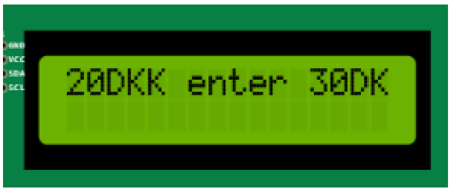


```
latte 50 00=1
01=5 10=10 11=20
```

Your prototype should accept "money" represented as the position of 2 **selection switches** for different coins with the following code:

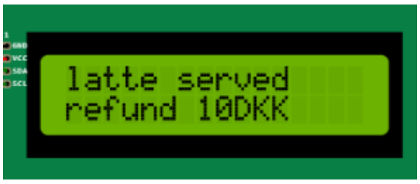
| **Amount** | **Amount Code** | |
|---|---|---|
| 1 DKK | 0 | 0 |
| 5 DKK | 0 | 1 |
| 10 DKK | 1 | 0 |
| 20 DKK | 1 | 1 |

In this example the user has entered 20 DKK and has to enter the remaining 30DKK by selecting the switches and pressing the enter push button every time.



```
20DKK enter 30DK
```

After the coffee product has been fully paid by inserting the “coins” the LCD will show a message saying that the coffee has been served and indicate if any refund amount needs to be returned to the user, in this example 10 DKK.



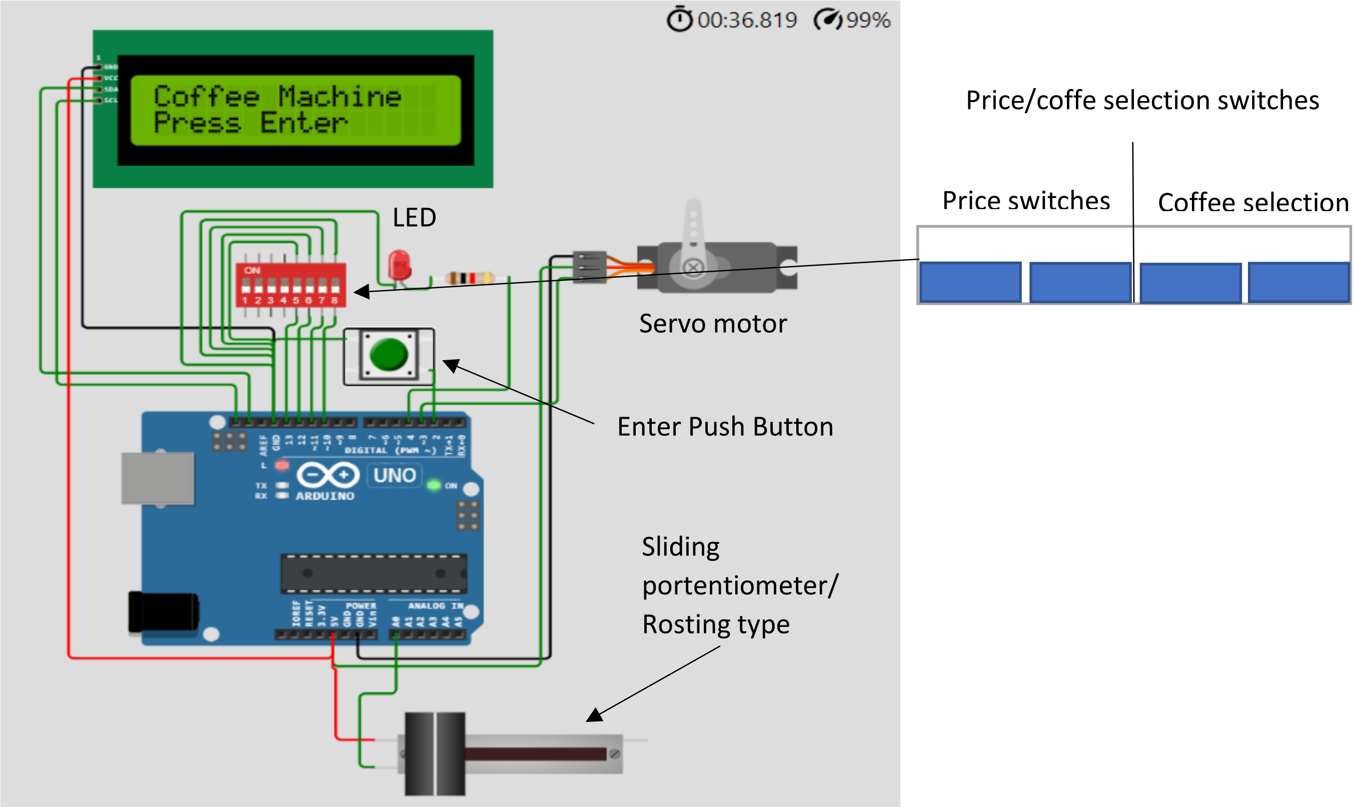
```
latte served
refund 10DKK
```

To actually “serve” the coffee, your prototype will have a servo motor that will move its shaft according to the type of coffee selected with the following degrees.

| Coffee Type | Servo Motor Angle |
|---|---|
| None | 0° |
| Regular | 45° |
| Espresso | 90° |
| Cappuccino | 135° |
| Latte | 180° |

After each purchase **both the servo motor should move the right angle and LED turn on for 5 seconds to indicate the coffee is ready**.

A  diagram of the HW of your coffee vending machine is shown below:



The diagram shows that the output of the user interface of your program will be a 16x2 I2C Liquid Crystal Display.

IMPORTANT: your hardware should use the same digital input/output ports, the same analog input and PWM port as shown in the diagram since I will test each of your programs in my simulator.

**Software Requirements:**

- You should use the **specific hardware shown on the diagram above i.e. port pins**. This is important since I will run your source code in my simulator.
- You should **NOT USE Arduino style code** but **only C++ style code for initialization, digital I/O, DAC and in all interrupts and timer**. To control LCD display and the servo motor you can use the Arduino libraries "LiquidCrystal_I2C.h" and "Servo.h". A description of the functions available in the libraries can be found at:
  https://www.arduino.cc/reference/en/libraries/liquidcrystal-i2c/
  https://www.arduino.cc/en/Reference/Servo
- You should **not use any delay() or _delay_ms() functions, to move the servo and turn on the LED 5 seconds. Your program should use Timer2 with interrupts to calculate a delay of 5 seconds to indicate when the coffee has been "served" by moving the servo the right amount of degrees shown on the table above. The only place where _delay_ms() function may be used is to debounce the enter push button.**
- Your program should be designed with **a finite state machine (FSM)**
- Your program should use **classes to represent useful objects for the problem domain i.e. a class for the whole coffee vending machine, another for the FSM. Optionally you may define more classes if you think they are useful in your prototype.**
- **Your program should use INT0 to connect the enter push button. The push button should be debounced in SW using this interrupt.**
- **Your program should use A0 ADC with interrupt to convert the voltage of the potentiometer into 4 values.**
- **You should use dynamic memory in your code i.e. pointers, but you may decide in which part to use them. Explain where and why you think the use of pointers makes sense in your specific design.**

**General requirements**

- The code should look neat
- **All functions in your source code should be well documented with header comments that describe what functionality they provide, what is the purpose of its arguments and if it returns a value what is it.**

Your program should fulfil all stated requirements as minimum, but you are welcome to include any other features that you may want to use to improve your design. **State in your miniproject report what you have added and how you had fulfilled all the requirements stated above.**

**IMPORTANT: If you do not fulfil all requirements stated or if you share your code with other groups, both groups may fail the miniproject.**