⟨*program*⟩ → ⟨*roots*⟩

⟨*letter*⟩ → [a - zA - Z]

⟨*digits*⟩ → ⟨*digit*⟩
| ⟨*digit*⟩ ⟨*digits*⟩

⟨*digit*⟩ → [0 - 9]

⟨*roots*⟩ → ⟨*root*⟩
| ⟨*root*⟩ ⟨*roots*⟩

⟨*root*⟩ → ⟨*function*⟩
| ⟨*dcl*⟩;
| ⟨*comment*⟩

⟨*dcl*⟩ → ⟨*type*⟩ ⟨*id*⟩
| ⟨*type*⟩ ⟨*assign*⟩

⟨*callid*⟩ → ⟨*id*⟩
| ⟨*callid*⟩[⟨*digits*⟩]

⟨*id*⟩ → ⟨*letter*⟩
| ⟨*id*⟩ ⟨*letter*⟩
| ⟨*id*⟩ ⟨*digit*⟩

⟨*assign*⟩ → ⟨*callid*⟩ <-- ⟨*expr*⟩

⟨*type*⟩ → ⟨*primitivetype*⟩
| ⟨*arraytype*⟩

⟨*primitivetype*⟩ → bool
| double
| int
| char
| container

⟨*arraytype*⟩ → ⟨*type*⟩ [ ]
| string

⟨*function*⟩ → function ⟨*id*⟩ return ⟨*type*⟩ using (⟨*params*⟩) begin ⟨*stmts*⟩ return ⟨*expr*⟩;
end
| function ⟨*id*⟩ return nothing using (⟨*params*⟩) begin ⟨*stmts*⟩ return nothing; end

⟨*params*⟩ → ⟨*subparams*⟩
| ε

⟨*subparams*⟩ → ⟨*type*⟩ ⟨*id*⟩, ⟨*subparams*⟩
| ⟨*type*⟩ ⟨*id*⟩

⟨*stmts*⟩ → ⟨*stmt*⟩
| ⟨*stmt*⟩ ⟨*stmts*⟩

⟨*stmt*⟩ → ⟨*assign*⟩;
| ⟨*if*⟩
| ⟨*while*⟩
| ⟨*from*⟩
| ε
| ⟨*dcl*⟩;
| ⟨*functioncall*⟩;
| ⟨*switch*⟩
| ⟨*comment*⟩

⟨*switch*⟩ → switch (⟨*expr*⟩) begin ⟨*cases*⟩ end

⟨*cases*⟩ → case ⟨*expr*⟩: ⟨*stmts*⟩ ⟨*endcase*⟩

⟨*endcase*⟩ → ⟨*cases*⟩
| break;
| break; ⟨*cases*⟩
| default: ⟨*stmts*⟩ break;
| break; default: ⟨*stmts*⟩ break;

⟨*expr*⟩ → ⟨*expr*⟩+⟨*term*⟩
| ⟨*expr*⟩-⟨*term*⟩
| ⟨*term*⟩

⟨*term*⟩ → ⟨*term*⟩ * ⟨*factor*⟩
| ⟨*term*⟩ / ⟨*factor*⟩
| ⟨*factor*⟩

⟨*factor*⟩ → (⟨*expr*⟩)
| ⟨*callid*⟩
| ⟨*plusminus*⟩⟨*digit*⟩
| ⟨*plusminus*⟩⟨*nummeric*⟩
| "⟨*string*⟩"
| ⟨*functioncall*⟩
| ⟨*cast*⟩
| LOW
| HIGH

⟨*plusminus*⟩ → ε
| -

⟨*nummeric*⟩ → ⟨*digit*⟩
| ⟨*digit*⟩⟨*nummeric*⟩
| .⟨*digitonly*⟩

⟨*digitonly*⟩ → ⟨*digit*⟩
| ⟨*digit*⟩ ⟨*digitonly*⟩

⟨*string*⟩ → ⟨*letter*⟩
| ⟨*digit*⟩
| ⟨*symbols*⟩
| ⟨*symbols*⟩ ⟨*string*⟩
| ⟨*digit*⟩ ⟨*string*⟩

| ⟨*letter*⟩ ⟨*string*⟩
| ε

⟨*symbols*⟩ → !
| %
| ^
| &
| *
| (
|)
| _
| +
| |
| ~
| -
| =
| `
| {
| }
| [
|]
| :
| ;
| ?
| ,
| .
| /
| ' '

⟨*from*⟩ → from ⟨*expr*⟩ to ⟨*logexpr*⟩ step ⟨*assign*⟩ begin ⟨*stmts*⟩ end

⟨*while*⟩ → while(⟨*logexpr*⟩) begin ⟨*stmts*⟩ end

⟨*if*⟩ → if(⟨*logexpr*⟩) begin ⟨*stmts*⟩ ⟨*endif*⟩

⟨*endif*⟩ → end else ⟨*if*⟩
| end else begin ⟨*stmts*⟩ end
| end

⟨*logexpr*⟩ → ⟨*logexpr*⟩ OR ⟨*andcomp*⟩
| ⟨*andcomp*⟩

⟨*andcomp*⟩ → ⟨*andcomp*⟩ AND ⟨*comp*⟩
| ⟨*comp*⟩

⟨*comp*⟩ → ⟨*booleanoperand*⟩ ⟨*comparisonoperator*⟩ ⟨*booleanoperand*⟩

⟨*booleanoperand*⟩ → true
| false
| ⟨*expr*⟩
| ⟨*boolean*⟩

⟨*boolean*⟩ → !(⟨*logexpr*⟩)
| (⟨*logexpr*⟩)

⟨*comparisonoperator*⟩ → >
| <
| <=
| >=
| !=
| =

⟨*functioncall*⟩ → call ⟨*id*⟩(⟨*callexpr*⟩)

⟨*callexpr*⟩ → ⟨*expr*⟩
| ⟨*expr*⟩, ⟨*callexpr*⟩

⟨*comment*⟩ → /* ⟨*string*⟩ */

⟨*cast*⟩ → ⟨*type*⟩ (⟨*expr*⟩)

0.1 Hardware

This section will be about the hardware components used in this project, describing them and reasons they are used in this project. In the description there be looked at the more basic technical specification that will be relevant for this project.

0.1.1 Arduino UNO

The Arduino UNO is a powerful microcontroller board which provides the user with ways to communicate with the microcontoller. An Arduino uses the ATmega328 chip which provides more memory than it predecessors and provides the Arduino board the computing power its known for [**?**].

The Arduino UNO board have 14 digital input/output pins where six of them can emulate an analogy output through PWM (Pulse-Width modulation) which are available on the Arduino board. The Arduino also provides the user with six analogy inputs which enables the reading of a alternating current and provides the user with the currents voltages. These pins can be used to control or perform readings other components and that way provides interaction with environment around the Arduino. The Arduino board is also mounted with an USB-port and jack socket. The board can be hooked up with an USB cable or a AC-to-DC (Alternating Current to Direct Current) adapter through the jack socket to power the unit. The Arduino UNO operates at 5v (volts) but the recommend range is 7-12v because lower current than 7v may cause instability if the unit needs to provide a lot of power to the attached modules. The USB is also used to program the unit with the desired program through a computer [**?**].

Programs for the Arduino are commonly made in Arduino's own language based on C and C++. The produces of the Arduino platform provides a development environment (Arduino IDE) that makes it possible to write and then simply upload the code to the connected Arduino platform. The produces also provides a library with functions to communicate with the platform and compatible components [**?**]. The Arduino is suited for this project because it makes it possible to demonstrate the language and illustrate that compilation is working. The Arduino is also a platform which is compatible with other modules and makes it possible to communicate and interact with them which also

is a desirable feature. These features are the reason why the Arduino platform have been chosen to work with in this project.

0.1.2 RFID

To administrate the users collection of purchased drinks the plan is to store the number and kind of drinks on a RFID tag that the customers then can use at the drink machine to get their drinks served.

RFID (Radio Frequency IDentification) are used to identify individual objects using radio waves. The communication between the reader and the RFID tag can go both ways, and you are able to read and write to most tag types. The objects able to be read differ a lot. It can be clothes, food, documents, pets, packaging and a lot of other kinds. All tags contains a unique ID that can in no way be changed from when they were made. This ID is used to identify an individual tag. Tags can be either passive to active tags. Passive tags do not do anything until their antenna catches a signal from a reader. This signal transfers enough energy to the tag for it to send a signal in return. active tags have a power source and therefore is able to send a signal on their own, making the read-distance greater. The tags can also be either *read only tag* or *read/write tag*. A *read only tag* only sends its ID back when it connects with a reader, while a *read/write tag* have a memory for storing additional information it then sends with the ID [**?**].

0.1.3 Other components

The demonstration situation will require something to illustrate more advance parts of theoretical machine. The plan is to use LEDs (light emitting diode) to illustrate the different function of the machine, when their are active or inactive. The LED is made of a semiconductor which produces a light when a current runs through the unit. LEDs are normally easy to use by simply run a current the correct way through the LED. The idea behind using LEDs is that there are not time for making the more advance parts of the machine nor is it the main focus of this project.

It is also desired that the is should be possible to print a form of text to the audience. To do this there will be used a LCD 16-pin (Liquid Crystal Display) which are compatible with the Hitachi44780 driver. Arduino's LiquidCrystal library provides the functions to write to LCD so that no low level code is needed to communicate with the LCD [**?**].

As input there will be used switches/buttons that will allow interaction with the program at runtime. The switches will illustrate a more advance control unit but in the project switches will be sufficient.

0.2 Token Specification

- bool
- int
- double
- char
- string
- $<--$
- OR

- AND
- <
- >
- <=
- >=
- !=
- =
- true
- false
- begin
- end
- if
- else
- function
- using
- return
- nothing
- switch
- case
- break
- default
- from
- to
- step
- while
- container
- motor