Thanh Huynh – 2688093

Bilal Butt – 2688700

Michel Clerger – 2694646

Zirui Song - 2652528

Design document

GROUP MEMBERS:

Contents

[CLASS DIAGRAM 2](#_Toc468452010)

[Class diagram 2](#_Toc468452011)

[Description of the classes and their members 2](#_Toc468452012)

[STATE DIAGRAMs 3](#_Toc468452013)

[PIN CONFIGURATIONS 3](#_Toc468452014)

# CLASS DIAGRAM

## Class diagram

We are working on. It will be updated.

## Description of the classes and their members

By default, all of the properties of classes are private and methods of them are public. If it is not the case, the visibilities of these properties and methods will be indicated.

By default, all of the member variables of classes are private and their methods are public. If it is not the case, the visibilities of these properties and methods will be indicated. The class diagram is still subject to refinement and will be amended as deemed necessary.

The classes as depicted in the class diagram are based on the notion of how the Arduino itself operates. Since the Arduino does not have an operating system instead it operates by consistently running a main loop that continues indefinitely. Therefore in the class hierarchy the LaundryMachine class plays the role of the class that contains the main loop.

The loop of the LaundryMachine class is constantly polling the ProgramSelect and CoinWallet classes to see if they have received any input from the user indicated by changes in their respective member variables. Once it has been determined if the program selected and the relevant payment has been completed and the user has indicated that he wants to begin the washing program, the ProgramExecutor will be called.

The ProgramExecutor executes the complete washing program for the washing program setting initially identified by the user. The interface classes are implemented in the HardwareControl class and are responsible for implementing functions that have to do with manipulating the actual hardware.

## HARDWARE RELATED CLASSES

#### ICOIN

ICoin is the interface class containing the virtual functions relating which is implemented in the hardware class.

|  |  |
| --- | --- |
| ICOIN | |
| Functions | Description |
| *virtual boolean GetCoin10Button() = 0;* | This is used for getting the status of “Coin 10” button which indicates that a coin of value 10 is put into the machine. |
|  |  |
|  |  |
| Remarks:  (Please describe here any problems, remarks, current states of your functions) | |
|  | |

#### ILOCK

ILock is the interface class containing virtual functions which are implemented in the hardware class.

|  |  |
| --- | --- |
| ILOCK | |
| Functions | Description |
| *virtual boolean GetLockStatus() = 0;* | This is used for getting the status of the switch belonging to the Door Lock on the simulator board. It returns a true if switch is turned on else it returns a false. It will only yield meaningful return values when keyselect is low. |
| Remarks: | |

#### ISOAP

ISoap is the interface class containing virtual functions which are implemented in the hardware class.

|  |  |
| --- | --- |
| ISOAP | |
| Functions | Description |
| *virtual boolean GetSoap1() = 0;* | This is used for getting the status of the switch belonging to the Soap 1 on the simulator board. It returns a true if switch is turned on else it returns a false. It will only yield meaningful return values when keyselect is low. |
| *virtual boolean GetSoap2() = 0;* | This is used for getting the status of the switch belonging to the Soap 2 on the simulator board. It returns a true if switch is turned on else it returns a false. It will only yield meaningful return values when keyselect is low. |
| *virtual void SetSoap1(boolean On) = 0;* | Will turn on the LED labelled Soap 1 on the simulation board if given argument true. Otherwise it will turn off LED for Soap 1 if given argument false. |
| *virtual void SetSoap2(boolean On) = 0;* | Will turn on the LED labelled Soap 2 on the simulation board if given argument true. Otherwise it will turn off LED for Soap 2 if given argument false. |
| Remarks: Function SetSoap2 still has some glitches because it is intended to be turned on by setting particular bits for Data and Group. However it seems to not be working despite using the prescribed method of deactivating and activating strobe in order to turn on or off the Soap 2 LED in question. | |

#### IWATER

IWATER is the interface class containing the virtual functions relating which is implemented in the hardware class.

|  |  |
| --- | --- |
| IWATER | |
| Functions | Description |
| *virtual void SetWaterLevel(int level) = 0;* | The purpose of this function will be to set the appropriate water lever for the washing process. It takes one parameter for the level that we want to check. |
| *virtual boolean CheckWaterLevel(int level) = 0;* | The purpose of this function will be to check if the washing machine has reached the desired water level. It takes one parameter for the level that we want to check. |
| *virtual void SinkWater() = 0;* | The purpose of this function will be to |
| Remarks: This interface is complete. All the functions are implemented.  To discuss: I just wonder if we need any more functions for this interface or are those 3 enough? | |

#### IMOTOR

IMOTOR is the interface class containing the virtual functions relating which is implemented in the hardware class.

|  |  |
| --- | --- |
| IMOTOR | |
| Functions | Description |
| *virtual void SetDirection(char dir) = 0;* | The purpose of this function will be to indicate the direction of the washing machine. It takes one parameter to allow us to indicate the direction that we want to set. |
| *virtual void SetSpeed(int level) = 0;* | The purpose of this function will be to indicate the speed of the washing machine. It takes one parameter to allow us to indicate the speed that we want to set. |
| *virtual void StopMotor() = 0;* | The purpose of this function will be to stop the motor. |
| *virtual void StartMotor() = 0* | The purpose of this function will be to start the motor at an initial state assuming that all the prerequisites are handled properly within the IProgram interface. |
| *virtual void CheckLoadingLevel(int level) = 0;* | The purpose of this function will be to check le loading level of the washing machine. It takes one parameter for the level that we want to check. |
| Remarks: This interface is complete. All the functions are implemented.  To discuss: To start the motor can we just set the initial speed, or we also should consider some other states? Should we take care of the prerequisites in this function of in the IProgram interface? | |

#### Itemperature

ITemperature is the interface class containing the virtual functions relating which is implemented in the hardware class.

|  |  |
| --- | --- |
| ICOIN | |
| Functions | Description |
| *virtual void SetTemperature(int level) = 0;* | To set the heater to heat to the specified temperature. |
| Remarks:  Implemented, not tested | |

#### IBuzzer

IBuzzer is the interface class containing the virtual functions relating which is implemented in the hardware class.

|  |  |
| --- | --- |
| ICOIN | |
| Functions | Description |
| *virtual void SetBuzzer(int ms) = 0;* | To let the buzzer make noise for a period of milliseconds. |
| Remarks:  Not implemented to the HardwareControl.cpp | |

# STATE DIAGRAMs

## PIN CONFIGURATIONS

From our independent investigations we tried to determine which buttons correspond to which exact input descriptors. In the hardware control class the constants used for each of the corresponding buttons and switches were as follows:

Keyselect must be HIGH in order to read button inputs.

Button 1: IN\_IN0 and IN\_IN3, pressing button 1 raises IN0 and IN3 to high  
Button 2: IN\_IN0, pressing button 2 raises IN0 to high  
Button 3: IN\_IN3, pressing button 3 raises IN3 to high  
Button 4: IN\_IN2, pressing button 3 raises IN2 to high  
Button 5: IN\_IN1, pressing button 3 raises IN1 to high  
Button 6: IN\_IN1, IN\_IN2 and IN\_IN3, pressing button 1 raises IN1, IN2 and IN3 to high

Keyselect must be LOW in order to read switch inputs.

Switch 2: IN\_IN0, pulling up switch 2 raises IN0 to high  
Switch 3: IN\_IN3, pulling up switch 3 raises IN3 to high  
Switch 4: IN\_IN1, pulling up switch 4 raises IN1 to high  
Switch 5: IN\_IN2, pulling up switch 5 raises IN2 to high

*Figure 1: Buttons and Switches on Laundry Machine*