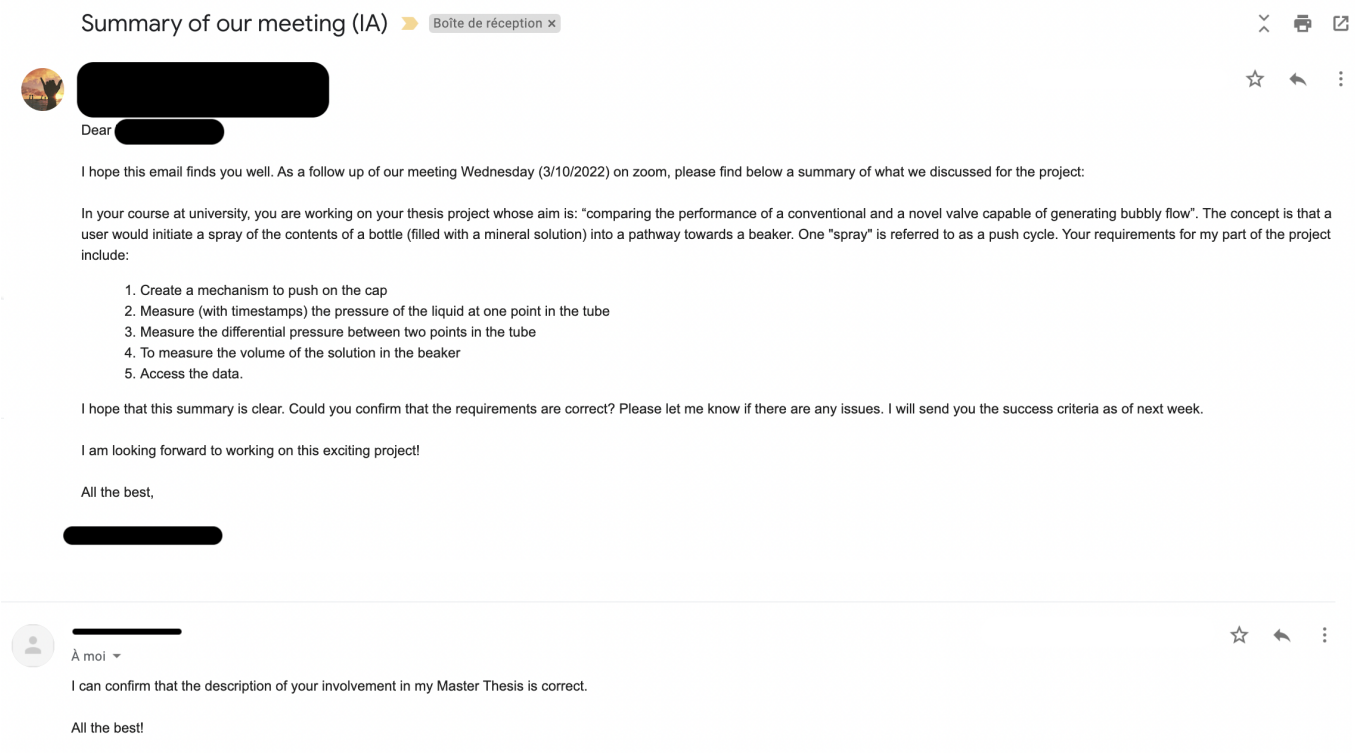
Appendix

1. Consultation with client

1.1. Consultation with client #1 (Meeting #1)



Summary of our meeting (IA) Boîte de réception ×

[redacted]

Dear [redacted]

I hope this email finds you well. As a follow up of our meeting Wednesday (3/10/2022) on zoom, please find below a summary of what we discussed for the project:

In your course at university, you are working on your thesis project whose aim is: "comparing the performance of a conventional and a novel valve capable of generating bubbly flow". The concept is that a user would initiate a spray of the contents of a bottle (filled with a mineral solution) into a pathway towards a beaker. One "spray" is referred to as a push cycle. Your requirements for my part of the project include:

1. Create a mechanism to push on the cap
2. Measure (with timestamps) the pressure of the liquid at one point in the tube
3. Measure the differential pressure between two points in the tube
4. To measure the volume of the solution in the beaker
5. Access the data.

I hope that this summary is clear. Could you confirm that the requirements are correct? Please let me know if there are any issues. I will send you the success criteria as of next week.

I am looking forward to working on this exciting project!

All the best,

[redacted]

[redacted]

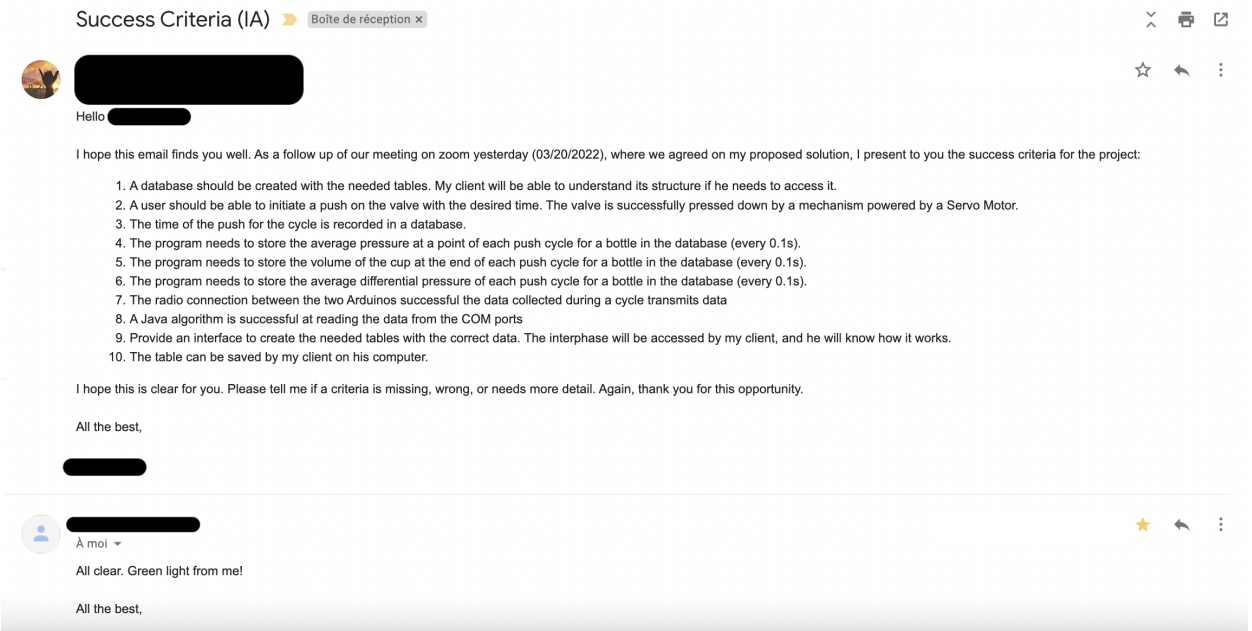
À moi

I can confirm that the description of your involvement in my Master Thesis is correct.

All the best!

Figure 1 - Evidence of meeting with client

1.2. Consultation with client #2 (Meeting #2)



Success Criteria (IA) Boîte de réception ×

[redacted]

Hello [redacted]

I hope this email finds you well. As a follow up of our meeting on zoom yesterday (03/20/2022), where we agreed on my proposed solution, I present to you the success criteria for the project:

1. A database should be created with the needed tables. My client will be able to understand its structure if he needs to access it.
2. A user should be able to initiate a push on the valve with the desired time. The valve is successfully pressed down by a mechanism powered by a Servo Motor.
3. The time of the push for the cycle is recorded in a database.
4. The program needs to store the average pressure at a point of each push cycle for a bottle in the database (every 0.1s).
5. The program needs to store the volume of the cup at the end of each push cycle for a bottle in the database (every 0.1s).
6. The program needs to store the average differential pressure of each push cycle for a bottle in the database (every 0.1s).
7. The radio connection between the two Arduinos successful the data collected during a cycle transmits data
8. A Java algorithm is successful at reading the data from the COM ports
9. Provide an interface to create the needed tables with the correct data. The interphase will be accessed by my client, and he will know how it works.
10. The table can be saved by my client on his computer.

I hope this is clear for you. Please tell me if a criteria is missing, wrong, or needs more detail. Again, thank you for this opportunity.

All the best,

[redacted]

[redacted]

À moi

All clear. Green light from me!

All the best,

Figure 2 - Evidence of meeting with client #2

1.3. Consultation with client #3

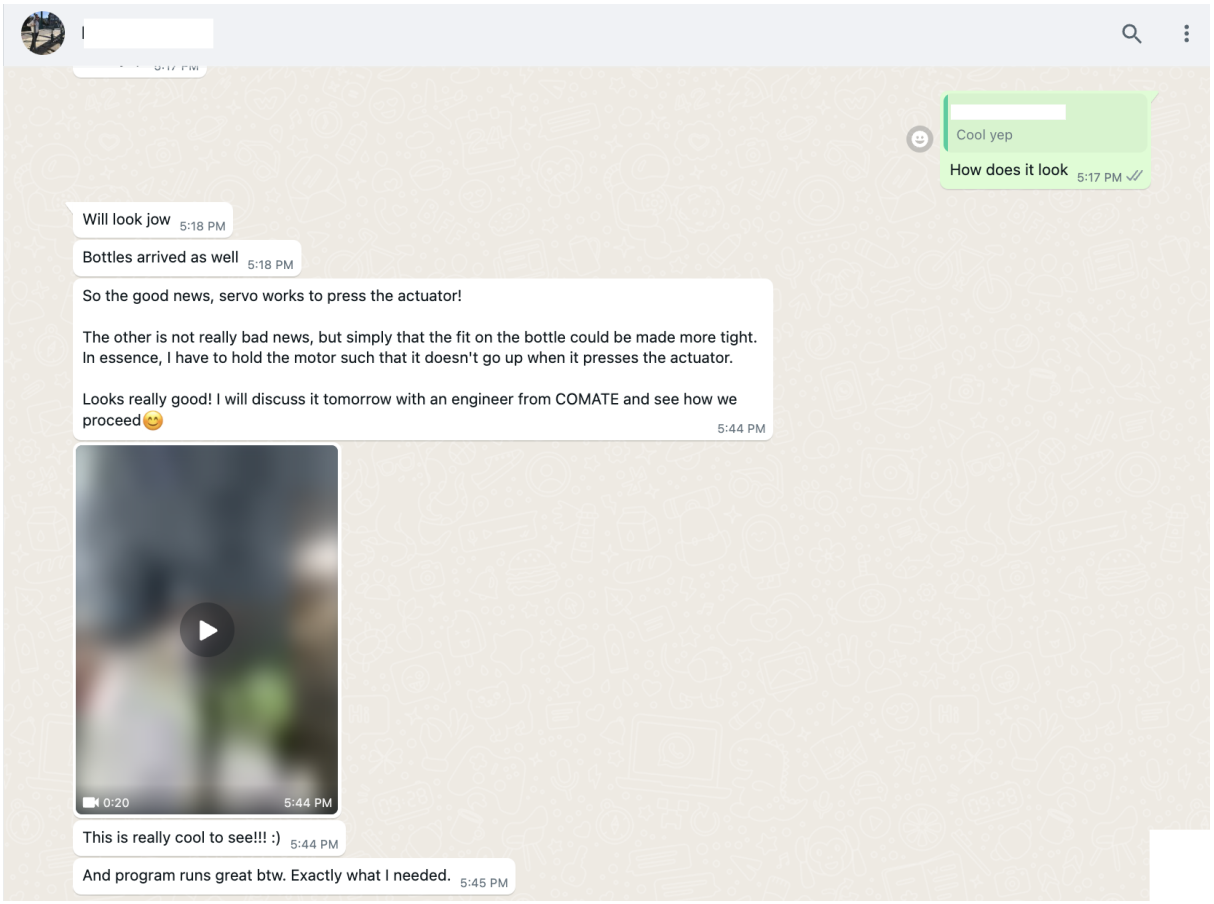


Figure 3 - Evidence of meeting Success Criteria #2.

My client and I kept close contact through Whatsapp, as he needed quick amends to the program that I was providing him. This is because he was conducting experimentation, and certain aspects of his methodologies changed along the process.

1.4. Consultation with client #4 (Meeting #3)

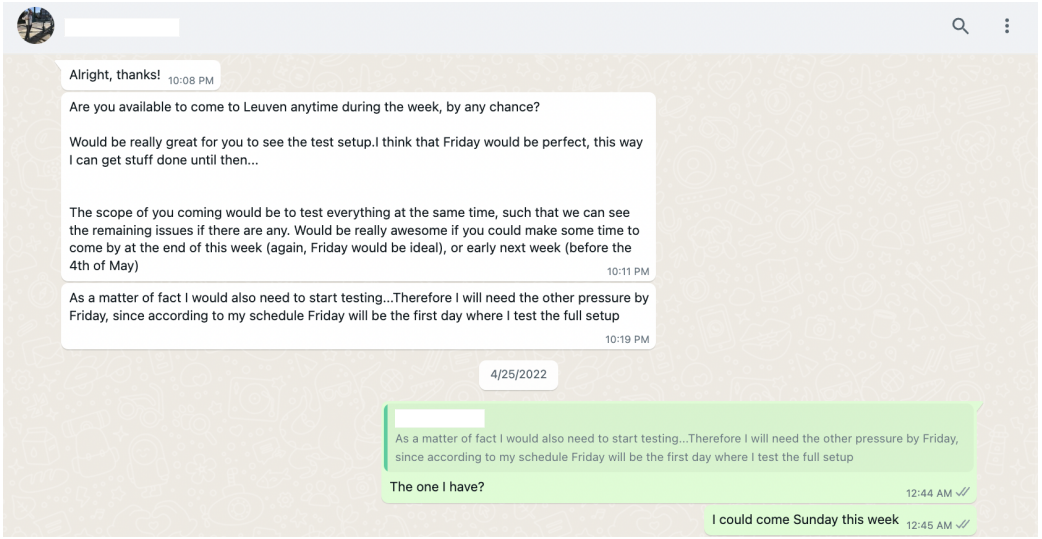


Figure 4 - Evidence of 3rd meeting with client for first testing at lab.

1.5. Consultation with client #5

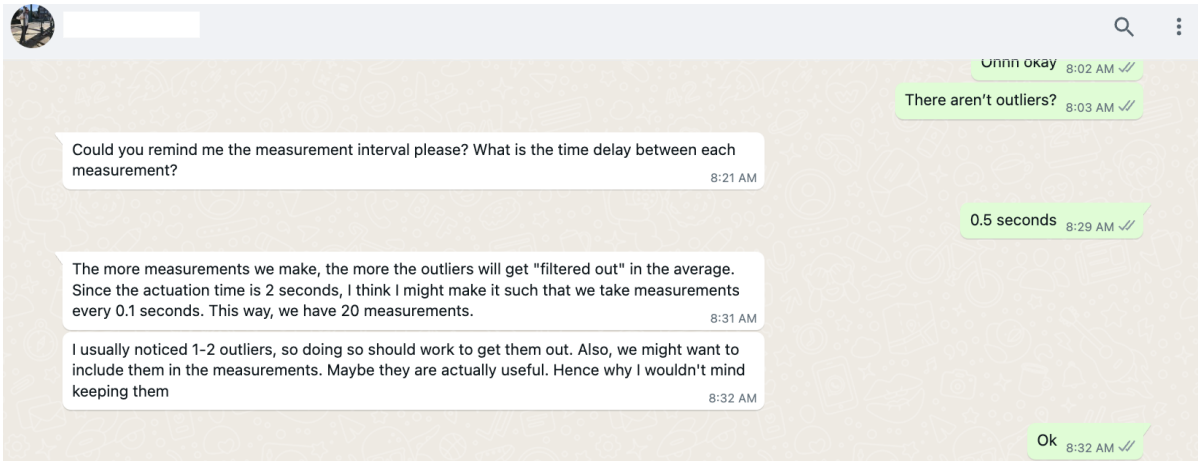


Figure 5 - Evidence of improving the prototype upon 3rd meeting and first testing.

1.6. Consultation with client #6 (Success criteria reached)

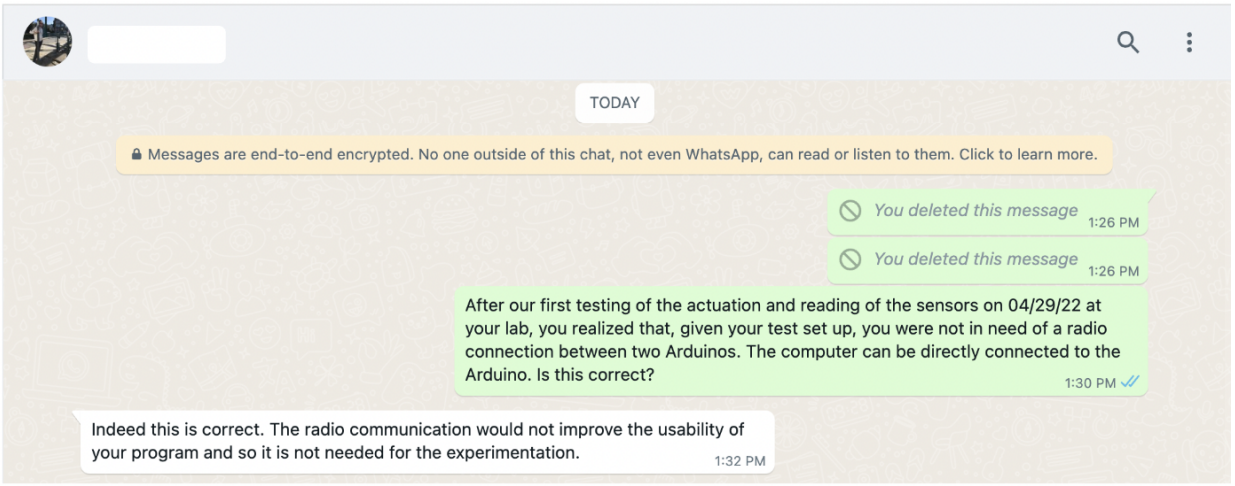


Figure 6 - Evidence for not using radio communication.

1.7. Consultation with client #6 (Success criteria reached)

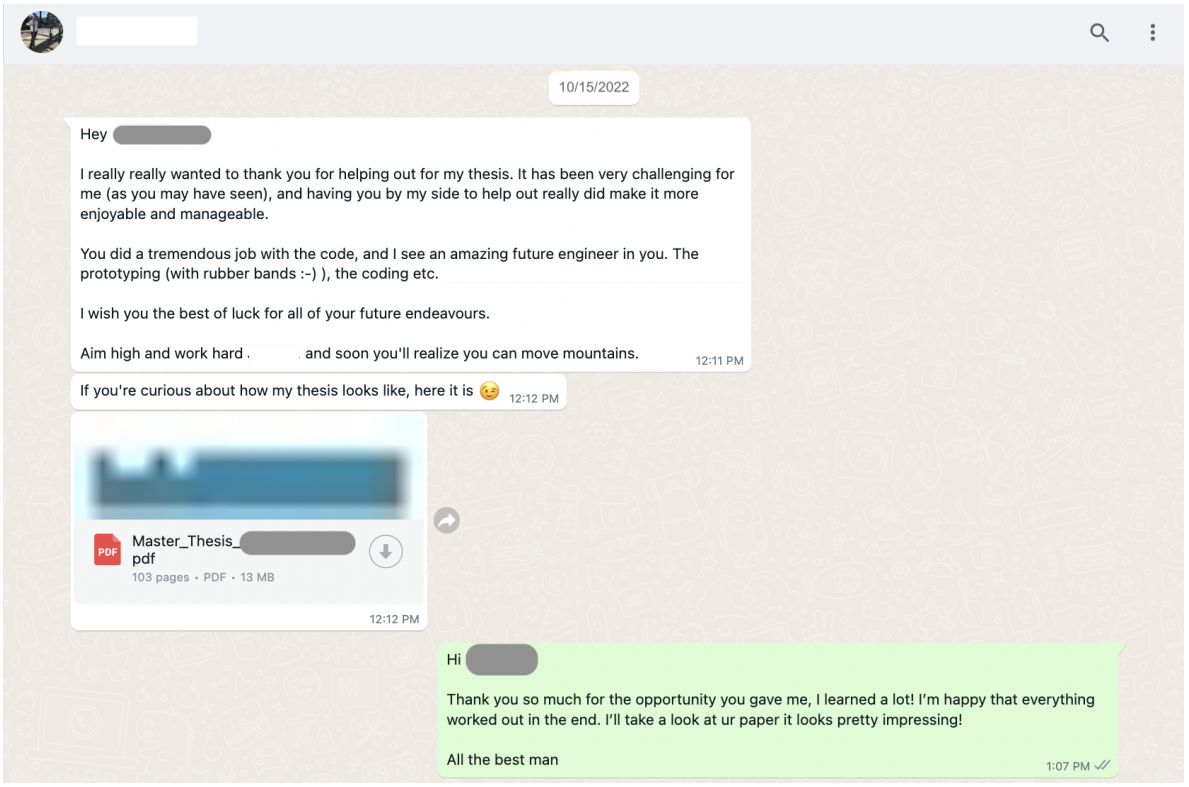


Figure 7 - Evidence of needed success criteria reached

1.8. Consultation with client #7 (Final meeting for future steps)

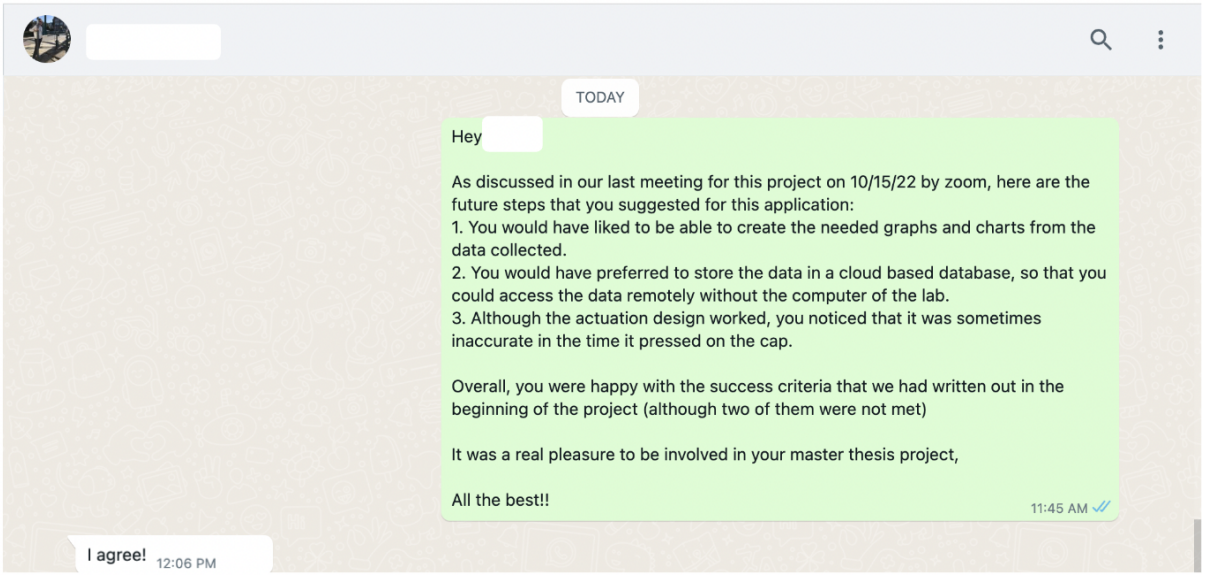


Figure 8 - Final meeting and consultation with client

2. Arduino code

Entirety of Arduino program

```
#include <Servo.h> //This library allows an Arduino board to control servo motors

  //Variables to action the servo motor for the actuationTime
   unsigned char servoState = LOW; //Stores the physical state of the servo
   unsigned char State = LOW; //Stores state of button that was pressed cycle
   unsigned char lastState = LOW; //Stores the previous state of button
   unsigned long servoCameOn = 0; //Stores the time that the servo came on

   unsigned long actuationTime = 1; //ACTUATION TIME OF THE CAP (in seconds)

   int push; //Stores the number of cycles
   boolean EXIT = false; //Enables to exit loop of the actuation of the servo

  int timer = (actuationTime*10); //Amount of measurements per cycle

  //Variable for sensors
  unsigned long previousMillis = 0;
  unsigned long Time;
  int interval1 = 100; //Constant time between measurements in cycles.

  //Array for to store the measurements of the volume
  float VolumeSensor[50];

  //Variables for the pressure measurements
  float AveragePress[50];
  float averageA1 = 0; //Stores the 1 pressure sensor

  //variables for differential pressure measurements
  float AverageDiff[50];
  float averageA2A3 = 0; //Stores the average of differential pressure

  int END = 1; //Used to end the arduino process once enough cycles have been done

int pos = 0; // variable to store the servo position

Servo myservo; //Creates a servo object called "myservo"

//--------------------------------------------------------------------------------
void setup() {
  //The setup program runs once:
  Serial.begin(9600); //readies the communication of the Serial Monitor at a data rate of 9600 bits per second
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object

}
//--------------------------------------------------------------------------------
void getTime(){

  //Get the number of seconds
  delay(100); Serial.println("Press NEXT for cycle or END to store measurements:"); //Prompt User for Input

  while (Serial.available() == 0) { //Loops until the user presses next

  }

  END = Serial.parseInt(); //END stores the value from the user input

  if(END == 9){
    //Serial.println("RECEIVED 9");
    FinalResults(); //calls the method to end the application
  }

  push = push + 1;
  State = HIGH;
  EXIT = false;

  Serial.println("-----------------------------------------------------");   delay(100);
  Serial.print("Cycle "); Serial.println(push);  delay(100);
  Serial.print("the cap will be pressed for "); Serial.print(actuationTime); Serial.println(" seconds");  delay(100);
  Serial.println("-----------------------------------------------------");
  delay(100);

  actuationTime = actuationTime * 1000; //This calculation is needed because the program counts in milliseconds
  }

//-------------------------------------------//-------------------------------------------//--------------------------

void serialFlush(){ //Method that clears the Serial.monitor
  while(Serial.available() > 0) {
    char t = Serial.read();
  }

}
```

```
//-----------------------------------------//-----------------------------------------//---------------------------

void ServoON(){
  do{
    // If the Servo has been on for longer than the actuation time seconds then turn it off.
    if(servoState == HIGH) //Checks if the servo is on
    {
      if(millis()-servoCameOn > actuationTime){ //Checks if the actuation time has passed

        myservo.write(0); //Resets the servo position
        servoState = LOW;
        EXIT = true;

        serialFlush(); //calls this method in order to clear the Serial. (avoid errors against false inputs)
        lastState = LOW;
        State = LOW;
      }
    }

    if(State != lastState) //If the state of button is the opposite of the previous push
    {
      lastState = State;
      if((State == HIGH) && (servoState == LOW)) //If the button has been pushed and the servo is at its original position
      {
        myservo.write(-180); //Servo is activated
        servoState = HIGH; //State of servo is high
        servoCameOn = millis(); //Know when the servo has turned

        //Method reads the sensors while the cap is pressed
        readSensors();
        myservo.write(0); //Resets the Servo to its original position
      }
    }
  }while(EXIT != true); //Loop while Exit = false
}




//-------------------------------------------------------------------------

void readSensors(){

  //pressure sensor that is by itself
  float PressAlone;
  long millisNow;
  //two pressure sensors
  float PressSensor1;
  float PressSensor2;
  float diffInPres;

  millisNow = millis(); //the time before the measurements

do{
      //if half 0.1 (value of interval1) seconds has passed → get measurements
      if (millis() - previousMillis >= interval1) {

        previousMillis = millis(); //stores the time when the measurements start

        //Stores the values of the pressure sensor that is alone A1
        PressAlone = analogRead(A1);
        averageA1 = averageA1 + PressAlone; //adds values for the average


        //Values of the for difference pressure, added and averaged
        PressSensor1 = (analogRead(A2));
        PressSensor2 = (analogRead(A3));
        diffInPres = (PressSensor2 - PressSensor1);
        averageA2A3 = averageA2A3 + diffInPres; //adds values for average
      }

 }while(millis()-servoCameOn < actuationTime-10);
 //Loop while the servo has been on for less than or equal to the set time

  //Stores the reading of the water sensor at the end of each cycle.
  VolumeSensor[push-1] = analogRead(A0);
}

//-------------------------------------------------------------------------

void CycleSummary(){
  //gets the average for the A1 sensor
  AveragePress[push-1] = (averageA1 / (timer));
```

```
  //gets the average for the difference in pressure A2 and A3
  AverageDiff[push-1] = (averageA2A3 / (timer));

  delay(200); Serial.println(); Serial.print("------------  Cycle ");
  Serial.print(push);  Serial.println(" Completed Successfully --------------");
  Serial.println(); delay(100);

  //puts back the timer to its original size after a cycle
  actuationTime = actuationTime/1000;
  averageA1 = 0;
  averageA2A3 = 0;

}

//-------------------------------------------------------------------------

void FinalResults(){

  String Vol = "";
  String Pres = "";
  String Diff = "";

  if(AveragePress[0] == 0){ //If there have not been any cycles, then loop again
    Serial.println("no cycles have been made");
    loop();
  }

  for(int c = 0; c < push; c++){    //Loop to create the Strings
      Vol = Vol + VolumeSensor[c] + " ";
      Pres = Pres + AveragePress[c] + " ";
      Diff = Diff + AverageDiff[c] + " ";
  }

  //Print the strings to the Serial monitor
  Serial.print("Pres");
  for(int c = 0; c < push; c++){    //Loop to create the Strings
    Serial.print(AveragePress[c]); Serial.print(" ");

  }

  Serial.print("Dif");
  for(int c = 0; c < push; c++){    //Loop to create the Strings
    Serial.print(AverageDiff[c]); Serial.print(" ");
  }

  Serial.print("|Vol");
  for(int c = 0; c < push; c++){    //Loop to create the Strings
   Serial.print(VolumeSensor[c]); Serial.print(" ");

  }
  Serial.println("?");

  delay(100);
  exit(0);
}

//-------------------------------------------//------------------------------

void loop() {

  getTime();
  ServoON();
  CycleSummary();

}
```

3. Java code

MainWindow.Java

```
import java.awt.EventQueue;

public class MainWindow{

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					LaunchWindow window = new LaunchWindow();
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

}
```

LaunchWindow.Java

```
import javax.swing.JFrame;
import javax.swing.JButton;
import java.awt.Color;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

public class LaunchWindow extends MainWindow {

	public static JFrame LW;

	/**
	 * Create the application.
	 */
	public LaunchWindow() {
		initialize();
	}

	/**
	 * Initialize the contents of the LW.
	 */
	private void initialize() {
		LW = new JFrame();
		LW.setBounds(100, 100, 450, 300);
		LW.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		LW.getContentPane().setLayout(null);

		JButton LoadTable = new JButton("Load Table"); //BUTTON TO LAUNCH LOAD TABLE MENU
		LoadTable.setForeground(new Color(248, 0, 7));
		LoadTable.addActionListener(new ActionListener() { //LoadTable Action Listener
			public void actionPerformed(ActionEvent e) {
				LW.dispose(); //Dispose the LW before creating a new instance
				LoadDataSava LD = new LoadDataSava(); //Create a new instance of the LoadDataSava class
			}
		});
		LoadTable.setBackground(new Color(99, 170, 255));
		LoadTable.setBounds(63, 108, 130, 41);
		LW.getContentPane().add(LoadTable); // Add to LW

		JButton GatherData = new JButton("Gather Data"); //BUTTON TO LAUNCH LOAD TABLE MENU
		GatherData.addActionListener(new ActionListener() { //GatherData Action Listener
			public void actionPerformed(ActionEvent e) {
				LW.dispose(); //Dispose the LW before creating a new instance
				new GatherDataMenu(); //Create a new instance of the GatherDataMenu
			}
		});


		GatherData.setBounds(248, 108, 130, 41);
		LW.getContentPane().add(GatherData); //Add to the LW

		JButton btnNewButton_2 = new JButton("help");
```

```
                    btnNewButton_2.addActionListener(new ActionListener() {
                              public void actionPerformed(ActionEvent e) {
                    JFrame f = new JFrame();
                    JOptionPane.showMessageDialog(f, "This application was designed to provide a digital solution to measuring and
storing data with sensors for an experiment"
                                        + "\nMy client was a Master student who was conducting experiments for his Master Thesis
project in chemical engineering"
                                        + "\nPress Gather Data to start your experiments"
                                        + "\nPress Load Table to create table from existing measurements");

                              }
                    });
                    btnNewButton_2.setForeground(Color.BLUE);
                    btnNewButton_2.setBounds(170, 211, 117, 29);
                    LW.getContentPane().add(btnNewButton_2);

                    JButton btnNewButton_2_1 = new JButton("exit");
                    btnNewButton_2_1.setForeground(Color.BLUE);
                    btnNewButton_2_1.setBackground(Color.LIGHT_GRAY);
                    btnNewButton_2_1.setBounds(170, 237, 117, 29);
                    LW.getContentPane().add(btnNewButton_2_1);

                    JLabel lblNewLabel = new JLabel("Welcome");
                    lblNewLabel.setBounds(195, 49, 61, 16);
                    LW.getContentPane().add(lblNewLabel);

                    LW.setVisible(true);
          }
}
```

GatherDataMenu.Java

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JTextArea;
import javax.swing.ScrollPaneConstants;
import javax.swing.SwingUtilities;
import javax.swing.text.DefaultCaret;

import com.fazecast.jSerialComm.*; // Serial Port classes

import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;

import java.awt.event.ActionListener;
import java.util.concurrent.TimeUnit;
import java.awt.event.ActionEvent;
import java.awt.Font;

public class GatherDataMenu extends Thread {

          static JFrame GDM;
          static JTextArea textArea = new JTextArea();
          static JButton ButStart = new JButton("Start"); //Button that starts the program
          static JButton ButNext = new JButton("Next"); //Button for next cycle
          static JButton ButEnd = new JButton("End"); //Button to end the program
          static JLabel lblNewLabel = new JLabel("Gather Data Menu");
          static JScrollPane taScroll;
          static SerialPort MySerialPort;

          static String S = "S"; //Variable for the transmission of data
          static boolean NextState = false;
          static boolean EndState = false;

          static String Results = ""; //Results of the bottle
          static int CounterCycle = 0;

          //Arrays to store values
          static String SPress[];
          static String SDi[];
          static String SVo[];

          /**
           * Create the application.
           */
          public GatherDataMenu() {
                    initialize();
```

```
	}

	/**
	 * Initialize the contents of the GDM.
	 */
	private void initialize() {
		GDM = new JFrame();
		GDM.setBounds(100, 100, 635, 450);
		GDM.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		GDM.getContentPane().setLayout(null);
		textArea.setFont(new Font("SansSerif", Font.PLAIN, 13));

		textArea.setLineWrap(true);
		textArea.setBounds(40, 36, 528, 320);
		textArea.setEditable(false);
		DefaultCaret caret = (DefaultCaret)textArea.getCaret();
		caret.setUpdatePolicy(DefaultCaret.ALWAYS_UPDATE);

		taScroll = new JScrollPane(textArea);
		taScroll.setBounds(40, 36, 528, 320);
		taScroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);

		GDM.getContentPane().add(taScroll);

		ButStart.setBounds(53, 368, 117, 29);
		GDM.getContentPane().add(ButStart);

		ButNext.setBounds(265, 368, 117, 29);
		ButNext.setEnabled(false);
		GDM.getContentPane().add(ButNext);

		ButEnd.setEnabled(false);
		ButEnd.setBounds(451, 368, 117, 29);
		GDM.getContentPane().add(ButEnd);

		lblNewLabel.setBounds(265, 18, 123, 16);
		GDM.getContentPane().add(lblNewLabel);

		ButStart.addActionListener(new ActionListener() { //ActionListener for ButStart
			public void actionPerformed(ActionEvent e) {
				BotStart.setEnabled(false); it cannot be pressed again
				new Thread(() -> {                          //Creating a new thread
					InitializeTransmission();
					GetData();
				}).start();
			}
		});

		GDM.setVisible(true);
	}

	public static void InitializeTransmission() {

			ButStart.setEnabled(false); //it cannot be pushed again
			//Variables for the Reception
			int BaudRate = 9600;
			int DataBits = 8;
			int StopBits = SerialPort.ONE_STOP_BIT;
			int Parity   = SerialPort.NO_PARITY;

			SerialPort [] AvailablePorts = SerialPort.getCommPorts(); //Initializes the SerialPort array object and storing the
available ports on the computer

		//Open the first Available port
		MySerialPort = AvailablePorts[AvailablePorts.length - 2]; //Specific to my computer

		// Set Serial port Parameters
		MySerialPort.setComPortParameters(BaudRate,DataBits,StopBits,Parity);//Sets all serial port parameters at one time
		MySerialPort.setComPortTimeouts(SerialPort.TIMEOUT_READ_BLOCKING, 1000, 0); //Set Read Time outs
		MySerialPort.openPort(); //open the port

		if (MySerialPort.isOpen()) { //If the Connection is open
			textArea.append("\nPORT:  " + MySerialPort.getSystemPortName() + " IS OPEN \n");
		}else {
				textArea.append("  ");
		   JFrame f = new JFrame();
		   JOptionPane.showMessageDialog(f, "ERROR: Arduino is not connected");
		   GDM.dispose();
		   GatherDataMenu LWD = new GatherDataMenu(); //Creates an new instance of the class if
		}

		   MySerialPort.flushIOBuffers();
		   System.out.println("Initialization complete");

		   //After Initialization is finished, exchange of data between Java and Arduino.
			boolean ENDLOOP = true;
			try {
```

```
                                        while (ENDLOOP == true){

                                                NextState = false;
                                                EndState = false;

                                                byte[] readBuffer = new byte[200]; //Creates a byte object because bytes are
read from the Serial monitor and then converted to strings
                                                int numRead = MySerialPort.readBytes(readBuffer, readBuffer.length); //Counts
the number of bytes in readBuffer
                                                S = new String(readBuffer, "UTF-8"); //convert bytes stored in readBuffer to
String
                                                textArea.append(S + "\n"); // Prints 'S' to the terminal

                                                int a = S.indexOf(":"); //Gets the index of ':' in String 'S'

                                                if(a != -1){ //If ':' is in the String S (meaning that user input has been
prompted)

                                                        ButNext.setEnabled(true); //buttons are enabled
                                                        if(CounterCycle != 0) {
                                                                ButEnd.setEnabled(true);
                                                        }

                                                        while((NextState == false) && (EndState == false)) { //While no
button has been pressed (this while loops freezes the program until user input

                                                                ButNext.addActionListener(new ActionListener() {
//ActionListener for Next button
                                                                        public void actionPerformed(ActionEvent e) {
                                                                                NextState = true;


                                                                        }
                                                                });

                                                                ButEnd.addActionListener(new ActionListener() {
//ActionListener for End button
                                                                        public void actionPerformed(ActionEvent e) {
                                                                                EndState = true;

                                                                        }
                                                                });
                                                        }

                                                        ButNext.setEnabled(false); //Buttons are disabled after they have
been pushed
                                                        ButEnd.setEnabled(false);

                                                        //If there has been 50 cycles then the program ends.
                                                        if(CounterCycle == 50) {
                                                                EndState = true; //This statement makes the loop end
                                                                NextState = false;
                                                                textArea.append("     Limit of cycles has been
reached\n");
                                                        }

                                                        //If End button has been pushed
                                                        if(EndState == true){
                                                                byte[] WriteByte = new byte[1]; //Byte to write to the
Serial Monitor
                                                                WriteByte[0] = 57; //send 9 to end the program (9 is the
argument that will trigger the end of the program in Arduino)
                                                                MySerialPort.writeBytes(WriteByte,1); //Method to write
element

                                                                TimeUnit.SECONDS.sleep(1); //Wait that the Serial Monitor
prints the final data
                                                                ENDLOOP = false; //Stop the loop
                                                        }


                                                        if(NextState == true){//SEND a byte so that Arduino thinks that
enter was pressed, so that the Arduino program runs

                                                                byte[] WriteByte = new byte[1];
                                                                WriteByte[0] = 65; //write an element to the serial
monitor, so that Arduino program continues running
                                                                MySerialPort.writeBytes(WriteByte,1);

                                                                a = -1; //Resets the condition for next if statement.
                                                                CounterCycle++; //The varibale that stores the number of
cycle increases
                                                        }
                                                }
                                        }

                                } catch (Exception e){ //Catch for try
```

```
                                        e.printStackTrace();
                                }

        }

        public static void GetData(){

                try{ //Try to get the results from the Serial monitor
                        byte[] readBuffer1 = new byte[1000];//Create a byte to retrieve the data
                        int numRead1 = MySerialPort.readBytes(readBuffer1, readBuffer1.length);
                        Results = new String(readBuffer1, "UTF-8"); //convert bytes to String

                } catch (Exception e) {
                        e.printStackTrace();
                }

      MySerialPort.flushIOBuffers();
      MySerialPort.closePort(); //close the Serial port

                //***Split 'Results' and store it in arrays

                //Get the indexes of characters in the String to split the strings into multiple subStrings
                int P1 = Results.indexOf("s");
                int P2 = Results.indexOf("f");
                int P3 = Results.indexOf("|");
                int P4 = Results.indexOf("?");

                //Create a substring from Results//FILE refers to the variable that stores the name of the database inputted by
user

                String TempPres = Results.substring(P1+1, P2-3);
                String TempDif = Results.substring(P2+1, P3-1);
                String TempVolume = Results.substring(P3+4, P4-1);

                //Store measurements in the arrays based on the substrings
                SPress = TempPres.split(" ");
                SDi = TempDif.split(" ");
                SVo = TempVolume.split(" ");

                textArea.append("Data was successfully imported to Java");
                GDM.dispose();
                System.out.println("GDM Disposed");
                SaveData sd = new SaveData(); //Create a new object of SaveData that will querry the data to a database
        }
}
```

SaveData.java

```
import java.awt.EventQueue;


import java.awt.event.ItemEvent;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JRadioButton;
import javax.swing.JComboBox;
import javax.swing.JTextField;
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

//For database
//For database connection
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
//import java.sql.Driver;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.awt.Font;
```

```
public class SaveData extends GatherDataMenu {

	static JFrame SD;
	static JComboBox comboBox;
	private JTextField textFieldDatabase;

	static String Model = "0";
	static String CycTime = "";
	static String Date =  "";

	//Variables for database
    static String MESSAGE;
    static Connection conn = null; //Sqlite data type for connections
	static String FILE;

    static int ID = 0;

	/**
	 * Create the application.
	 */
	public SaveData() {
		initialize();
	}

	/**
	 * Initialize the contents of the SD.
	 */
	private void initialize() {

		GDM.dispose();
		SD = new JFrame();
		SD.setBounds(100, 100, 299, 394);
		SD.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		SD.getContentPane().setLayout(null);

		JLabel lblNewLabel = new JLabel("Save Data");
		lblNewLabel.setFont(new Font("Lucida Grande", Font.PLAIN, 26));
		lblNewLabel.setBounds(85, 19, 132, 37);
		SD.getContentPane().add(lblNewLabel);

		JLabel lblNewLabel_1 = new JLabel("Fill these required fields");
		lblNewLabel_1.setBounds(29, 68, 162, 16);
		SD.getContentPane().add(lblNewLabel_1);

		JRadioButton Conventional = new JRadioButton("Conventional Valve");
		Conventional.setBounds(50, 86, 162, 23);
		SD.getContentPane().add(Conventional);

		JRadioButton Novel = new JRadioButton("Novel Valve");
		Novel.setBounds(50, 111, 141, 23);
		SD.getContentPane().add(Novel);

		ButtonGroup bg = new ButtonGroup();
		bg.add(Conventional); bg.add(Novel);

		String num[] = {"1", "2" , "3", "4", "5", "6", "7", "8", "9", "10"};
		comboBox = new JComboBox(num);
		comboBox.setBounds(51, 179, 72, 27);
		SD.getContentPane().add(comboBox);

		JLabel lblNewLabel_1_1 = new JLabel("Select Cycle Time");
		lblNewLabel_1_1.setBounds(29, 151, 162, 16);
		SD.getContentPane().add(lblNewLabel_1_1);

		JRadioButton CreateTable = new JRadioButton("Create Table now");
		CreateTable.setBounds(29, 290, 162, 23);
		SD.getContentPane().add(CreateTable);

		textFieldDatabase = new JTextField();
		textFieldDatabase.setColumns(10);
		textFieldDatabase.setBounds(50, 252, 130, 26);
		SD.getContentPane().add(textFieldDatabase);

		JButton btnNewButton = new JButton("Finish"); //Button cannot be on until all of the fields are filled.

		btnNewButton.setEnabled(false);
		textFieldDatabase.addKeyListener(new KeyAdapter() { //Key listener to check whether all required fields are
compeleted
	public void keyReleased(KeyEvent e) {
	    super.keyReleased(e);

			if(bg.getSelection() == null && textFieldDatabase.getText().length() == 0) {
				btnNewButton.setEnabled(false);
			}else {
				btnNewButton.setEnabled(true);
```

```
                        }
                    }
                });


                btnNewButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {

                                //Checks which button of the radioButton is checked
                                if (Conventional.isSelected()) Model = "0";
                                if (Novel.isSelected()) Model = "1";

                                CycTime = comboBox.getSelectedItem().toString(); //Get ComboBox chosen number number

                                FILE = textFieldDatabase.getText(); //Get name of the Database

                                SelectDatabase();
                                SendData();


                                if(CreateTable.isSelected()) {
                                        SD.setVisible(false);
                                        CreateTable newTable = new CreateTable();

                                } else {
                                        SD.dispose();
                                        GDM.dispose();
                                        MainWindow ms = new MainWindow();

                                }

                        }
                });
                btnNewButton.setBounds(85, 325, 117, 29);
                SD.getContentPane().add(btnNewButton);

                JLabel NameDatabase = new JLabel("Enter the name of the database");
                NameDatabase.setBounds(29, 234, 206, 16);
                SD.getContentPane().add(NameDatabase);


                SD.setVisible(true);
        }


        private static void openConn() {  //Need to have JDBC driver referenced.

                try {
         //Connect to the database
        //FILE refers to the variable that stores the name of the database inputted by user
         conn = DriverManager.getConnection("jdbc:sqlite:" + FILE); //Stores the connection created in 'conn'

                            //Connection is created in the same directory as the Java project
         MESSAGE = "DB \"" + FILE + "\" selected.";

    } catch (SQLException EX) {
         MESSAGE = "exception: " + EX; //Catch an error
        JFrame f = new JFrame();
        JOptionPane.showMessageDialog(f, "ERROR: try again"); //Error message
         SaveData sd = new SaveData(); //Create a new instance

    }
}

/*
* This method closes the connection. It should be called by all methods who open a DB connection
*/
private static void closeConn() {
    try {
        if (conn != null) conn.close(); //If a conn has been created then close the conn

    } catch (SQLException EX) { //If SQL catches an error
        if ((EX.getErrorCode() == 50000) && ("XJ015".equals(EX.getSQLState()))) {
            System.out.println("SQLite shut down properly");

        } else {
            System.err.println("SQLite did not shut down properly");
            System.err.println(EX.getMessage());
            closeConn(); //Call the method again to shut down the connection

        }

    }

}
```

```
    private static void SelectDatabase() {

           Statement stmt; //Variable that is used to execute SQL statement
        String strSQL = ""; //Variable to write the SQL statement in String

                    openConn(); //Open connection to current database

        // Open connection to current DB
        System.out.println(MESSAGE);

        // Try to create the table "tBottle" IF it doesn't already exist
        try{
            stmt = conn.createStatement();
            strSQL = "CREATE TABLE IF NOT EXISTS tBottle ("
                                                    + "Bot_ID INTEGER PRIMARY KEY AUTOINCREMENT,"
                                                    +" Bot_Model INTEGER NOT NULL, Bot_CycTime INTEGER NOT NULL, Bot_Date TEXT NOT
NULL);";
            stmt.execute(strSQL);
            System.out.println("Table tBottle is ready");

        } catch (SQLException EX) {
            System.out.println(EX.getMessage());

        }

        // Try to create the table "tCycles" IF it doesn't already exist
        try{
            stmt = conn.createStatement();
            strSQL = "CREATE TABLE IF NOT EXISTS tCycle ("
            + " Cycle_ID integer PRIMARY KEY AUTOINCREMENT,"
            + " Cycle_Description Text);";
            stmt.execute(strSQL);
            System.out.println("Table tCycle is ready");

           //Send tCycle
           final String sqlCyc = "INSERT INTO tCycle VALUES(?,?)"; //Prepare statemnt
                    for(int c = 1; c <= 50; c++){

                                            PreparedStatement pstmt = conn.prepareStatement(sqlCyc); //An object that represents a
precompiled SQL statement

                                            pstmt.setInt(1, c);
                                            pstmt.setString(2, null);
                                            pstmt.executeUpdate();
                                            pstmt.close();
                    }

        } catch (SQLException EX) {
            System.out.println(EX.getMessage());
        }

        // Try to create the table "tMeasurements" IF it doesn't already exist
        try{
            stmt = conn.createStatement();
            strSQL = "CREATE TABLE IF NOT EXISTS tMeasurements("
            + " Mes_ID integer PRIMARY KEY AUTOINCREMENT," //Create the primary key column
            + " Mes_Press Real not null, Mes_Diff Real not null, Mes_Disch Real not null, Mes_Vol Real not null," //Create needed
tables with datatype
            + " MesBot_ID integer not null, MesCyc_ID integer not null," //Create tables that will be used for foreign keys
            + " FOREIGN KEY(MesBot_ID) REFERENCES tBottle(Bot_ID)," //Create a reference to foreign keys
            + " FOREIGN KEY(MesCyc_ID) REFERENCES tCycle(Cycle_ID));"; //Create a reference to foreign keys

            stmt.execute(strSQL); //execute the SQLite statement
            System.out.println("Table tMeasurements is ready"); //feeback

        } catch (SQLException EX) {
            System.out.println(EX.getMessage());

        } finally {
           // Close the connection to database
           closeConn();
        }

    }

    public static void SendData() {

           openConn();
           //SEND tBOTTLE
           final String sqlBot = "INSERT INTO tBottle VALUES(?,?,?, date('now', 'localtime'))"; //Prepare statemnt
                    try{
                                            PreparedStatement pstmt = conn.prepareStatement(sqlBot); //An object that represents a
precompiled SQL statement

                                            pstmt.setString(1, null); //Can send as string and will be checked as REAL in the database
                                            pstmt.setString(2, Model);
```

```
                                        pstmt.setString(3, CycTime);
                                        pstmt.executeUpdate();
                                        pstmt.close();
                    }

                    catch (SQLException e){
                                        System.out.println("Error: " + e.getMessage());
                    }

          closeConn();
          openConn();
          //Insert measurements into tMeasurements
          String sqlMeas = "INSERT INTO tMeasurements VALUES(?,?,?,?,?,?,?)"; //Initialize a sql prepare statemnt

          try{
                     Statement stmt;
                ResultSet rs = null;
                String strSQL1 = "";
                System.out.println("Query done");

                strSQL1 = "SELECT Bot_ID " //This SQL script will retrieve the ID of the bottle that was just created.
                            + "FROM tBottle "
                            + "WHERE Bot_ID = (SELECT MAX(Bot_ID) FROM tBottle)";
                stmt = conn.createStatement();
                rs = stmt.executeQuery(strSQL1); //RS with the executed query

                if (rs.next()) { //retrieve the ID from the resultset
                    ID = rs.getInt("Bot_ID");
                }

                   for(int c = 0; c < CounterCycle; c++){ //Loop to Insert measurements stored in arrays in the database

                                        PreparedStatement pstmt = conn.prepareStatement(sqlMeas); //An object that represents a
precompiled SQL statement

                                        //The integer in the arguments of 'setString' refers to the index of the question mark in
the prepared statement

                                        pstmt.setString(1, null);     //Data is sent as string and translated to REAL according to
the datatype of the columns in the database
                                        pstmt.setString(2, SPress[c]); //Pressure
                                        pstmt.setString(3, SDi[c]);//Differential pressure
                                        pstmt.setString(4, SDi[c]);//Discharge rate
                                        pstmt.setString(5, SVo[c]);//Volume
                                        pstmt.setInt(6, ID); //BotID
                                        pstmt.setInt(7, c+1); //Cycle#

                                        pstmt.executeUpdate(); //Execute statement
                                        pstmt.close(); //close statement to recreate one for each iteration
                    }


          } catch (SQLException e){
          System.out.println("Error: " + e.getMessage());
                    }

          closeConn();
    }
}
```

CreateTable.java

```
import java.awt.Color;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JTable;
import javax.swing.KeyStroke;
import javax.swing.JButton;
import javax.swing.JComponent;

import java.awt.event.ActionListener;
import java.awt.event.InputEvent;
import java.awt.event.KeyEvent;
import java.awt.event.ActionEvent;
import javax.swing.JLabel;
import javax.swing.JScrollPane;

public class CreateTable extends SaveData {
```

```
	private static JFrame frame;
	static private JTable table;

	/**
	 * Create the application.
	 */
	public CreateTable() {
		initialize();
	}

	/**
	 * Initialize the contents of the frame.
	 */
	private void initialize() {
		SD.dispose();
		frame = new JFrame();
		frame.setBounds(100, 100, 549, 372);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.getContentPane().setLayout(null);

		Table(); //Initialize table

		JButton Finish = new JButton("Finish and Menu");
		Finish.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				frame.dispose();
				LaunchWindow LKW = new LaunchWindow();
			}
		});
		Finish.setBounds(205, 298, 137, 29);
		frame.getContentPane().add(Finish);

		if(Model.compareTo("0") == 0) {
			Model = "Conventional";
		} else {
			Model = "Novel";
		}

		JLabel Description = new JLabel(Model + " - " + CycTime + " seconds" + " - Bot_ID: " + ID);
		Description.setBounds(175, 23, 368, 16);
		frame.getContentPane().add(Description);

		JScrollPane scrollPane = new JScrollPane(table);
		scrollPane.setBounds(51, 64, 442, 222);
		frame.getContentPane().add(scrollPane);

		frame.setVisible(true);

	}

	public static void Table() {

		String [] ColumnNames = {"Cyc_ID", "Pressure", "Differential", "Discharge rate", "Volume"}; //String array for
the names of the columns
		String Data [][] = new String[CounterCycle][5]; //2D array that stores the Data used to create the JTable

		//CounterCycle is the amount of cycles, hence measurements, for a specific bottle

		//initialize the array with 5 column because there are 5 different variable that are displayed.

		for(int c = 0; c < CounterCycle; c++) { //Loop to put data in 2D array

			Data[c][0] = Integer.toString(c + 1);
			Data[c][1] = SPress[c];
			Data[c][2] = SDi[c];
			Data[c][3] = SDi[c];
			Data[c][4] = SVo[c];
		}

		table = new JTable(new MyTableModel(Data, ColumnNames)); //Initialize JTable with a table model that makes cell
not editable but selectable
		table.setCellSelectionEnabled(true);
		table.setBackground(Color.pink);
		ExcelAdapter myAd = new ExcelAdapter(table); //Class that enables copy and paste for the table
		table.setBounds(70, 50, 422, 226);

	}

}
```

LoadDataSava.java

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.File;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JComboBox;
import javax.swing.JFileChooser;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.ListSelectionModel;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import javax.swing.JButton;

public class LoadDataSava {

	private JFrame frame;
	static boolean showRest = false;
	static private JTable TempTable;

	//For Database SQL
	static String CMD = null;
    static String[] ARG;
    static String curDB;
    static String MESSAGE;
    static Connection conn = null;
	static String FILE;
	private JTextField textFieldDatabase;

	String row = "";
	static int response = 1;

	//For Table
	static String Table[][];
	static String[] ColumnNames = new String[4];

	static String Parameter = ""; //Search Parameter
	static int selectedRow;
	static int column;

	static public String Bot_ID;
	static public String Model;
	static public String Date;
	static public String CycTime;


	public LoadDataSava() { //Constructor
		initialize();
	}

	/**
	 * Initialize the contents of the frame.
	 */
	private void initialize() {
		frame = new JFrame();
		frame.setBounds(100, 100, 669, 484);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.getContentPane().setLayout(null);

		JLabel lblNewLabel = new JLabel("Load Data");
		lblNewLabel.setBounds(300, 23, 69, 16);
		frame.getContentPane().add(lblNewLabel);

		JLabel lblNewLabel_1 = new JLabel("Search by");
		lblNewLabel_1.setBounds(208, 87, 69, 16);

		JButton btnCreate = new JButton("Create Table"); //IF button pressed, get the *Bot_ID* and the tBottle to create table based on that.
		btnCreate.setBounds(33, 408, 117, 29);
		btnCreate.addActionListener(new ActionListener() {
```

```
                                public void actionPerformed(ActionEvent e) {
                                        if(e.getSource()==btnCreate) {
                                                column = TempTable.getColumnModel().getColumnIndex("Bot_ID");
                                                Bot_ID = TempTable.getValueAt(selectedRow, column).toString(); //Get Bot_ID

                                                column = TempTable.getColumnModel().getColumnIndex("Bot_Date");
                                                Date = TempTable.getValueAt(selectedRow, column).toString(); //Get Bot_Date

                                                column = TempTable.getColumnModel().getColumnIndex("Bot_CycTime");
                                                CycTime = TempTable.getValueAt(selectedRow, column).toString(); //Get
Bot_CycTime

                                                column = TempTable.getColumnModel().getColumnIndex("Bot_Model");
                                                Model = TempTable.getValueAt(selectedRow, column).toString(); //Get Bot_Model

                                                frame.dispose();
                                                LoadTable LT = new LoadTable();
                                        }
                                }
                        });

                        String num[] = {"None", "Date", "Bot_ID" , "Bot_Model", "Bot_CycTime"};
                        JComboBox comboSearch = new JComboBox(num);
                        comboSearch.setBounds(281, 83, 105, 27);

                        //Button to open database
                        JButton btnOpenDatabase = new JButton("Select Database");
                        btnOpenDatabase.setBounds(265, 42, 139, 29);
                        frame.getContentPane().add(btnOpenDatabase);

                        btnOpenDatabase.addActionListener(new ActionListener() { //Action listener for database button
                                public void actionPerformed(ActionEvent e) {
                                        if(e.getSource()==btnOpenDatabase) {

                                                //Using a JFile chooser to choose the DATABASE
                                                JFileChooser FC = new JFileChooser(); //Open file chooser
                                                FC.setCurrentDirectory(new File("."));
                                                response = FC.showOpenDialog(null); //variable to check the responde of the
file chooser

                                                if(response == JFileChooser.APPROVE_OPTION) { //If the approve option has been
selected
                                                        FILE = FC.getSelectedFile().getName();  //Store the name of the file
selected

                                                        JFrame f = new JFrame();
                                                JOptionPane.showMessageDialog(f, "Database Selected: " + FILE);  //Option Pane to say
that Database was selected.

                                                        frame.getContentPane().add(comboSearch);  //When Database has been
selected, other elements are placed on the GUI
                                                        frame.getContentPane().add(lblNewLabel_1);
                                                        frame.getContentPane().add(btnCreate);

                                                }
                                                btnOpenDatabase.setText(FILE); //The button now displaying the name of database
selected.
                                        }
                                }
                        });

                        ActionListener cbActionListener = new ActionListener() {
                                @Override
                                public void actionPerformed(ActionEvent e) {

                                        Parameter = (String) comboSearch.getSelectedItem();
                                        System.out.println(Parameter);

                                        Search();

                                        JScrollPane scrollPane = new JScrollPane(TempTable); //Create the scrollPane that is used
for search parameter
                                        scrollPane.setBounds(75, 122, 519, 261);
                                        scrollPane.setVisible(true);
                                        frame.getContentPane().add(scrollPane);


                                        TempTable.addMouseListener(new MouseAdapter() { //Mouse listener to check the row that was
clicked
                                            public void mouseClicked(MouseEvent e) {
                                                    selectedRow = TempTable.getSelectedRow();
                                                    System.out.println(selectedRow);
                                            }
                                        });

                                }
```

```
                    };
                    comboSearch.addActionListener(cbActionListener);

                    frame.setVisible(true);

          }

          /*
    * This method opens the connection.     */
          private static void openConn() {
        try {

           // Try to connect to the database
           conn = DriverManager.getConnection("jdbc:sqlite:" + FILE); //Need to have JDBC driver referenced.
           MESSAGE = "DB \"" + FILE + "\" selected.";
           System.out.println(MESSAGE);

        } catch (SQLException EX) {
           MESSAGE = "exception: " + EX;

        }
    }

    /*
    * This method closes the connection. It should be called by all methods who open a DB connection
    */
    private static void closeConn() {
        try {
           if (conn != null) conn.close();

        } catch (SQLException EX) {
           if ((EX.getErrorCode() == 50000) && ("XJ015".equals(EX.getSQLState()))) {
               System.out.println("SQLite shut down properly");

           } else {
               System.err.println("SQLite did not shut down properly");
               System.err.println(EX.getMessage());

           }

        }

    }

          public static void Search() {
                    openConn();

                    try{
                     Statement stmt;
                     Statement stmt1;
                  ResultSet rs = null;
                  ResultSet rowNum = null;

                  String strSQL = "";

                  switch(Parameter) { //Switch statement that selects the SQL script depending on the search paramenter of the
user.
                                                          //Parameter is the option chosen from the comboBox
                    case("Date"):
                                          // Select all from the bottle with the Date as the reference column
                                          strSQL = "SELECT Bot_Date, Bot_ID, Bot_Model, Bot_CycTime "
                                                      + " FROM tBottle"
                                                      + " ORDER BY Bot_Date ASC;";

                                case("Bot_ID"):
                                  // Select all from the bottle with the Bot_ID as the reference column
                                           strSQL = "SELECT Bot_ID, Bot_Date, Bot_Model, Bot_CycTime "
                                                                  + " FROM tBottle"
                                                                  + " ORDER BY Bot_ID ASC;";

                                case("Bot_Model"):
                                  // Select all from the bottle with the Bot_Model as the reference column
                                    strSQL = "SELECT  Bot_Model, Bot_ID, Bot_Date, Bot_CycTime "
                                                                  + " FROM tBottle"
                                                                  + " ORDER BY Bot_Model ASC;";

                                case("Bot_CycTime"):
                                  // Select all from the bottle with the Bot_CycTime as the reference column
                                    strSQL = "SELECT Bot_CycTime, Bot_ID, Bot_Date, Bot_Model "
                                                                  + " FROM tBottle"
                                                                  + " ORDER BY Bot_CycTime ASC;";
                  }

                  stmt = conn.createStatement(); //Allign the statement with the SQLite connection
                  System.out.println(strSQL);
                  rs = stmt.executeQuery(strSQL); //Execute the query
```

```
                    //Create a new query to store the number of rows in tBottle. This number will be used to initialize the number of
rows in the table (2D array).
                    String strSQL1 = "SELECT COUNT(*) FROM tBottle";
                    stmt1 = conn.createStatement();
                    rowNum = stmt1.executeQuery(strSQL1);

                    //Initialize 2D array for the table that will be used to search the needed bottle
                    Table = new String[(((Number) rowNum.getObject(1)).intValue())][4]; //Intialize array with correct numebr

                    ResultSetMetaData rsmd = rs.getMetaData(); //Using a resultSetMetaData object that provides information about the
columns in a ResultSet.

                    for(int c = 0; c < rsmd.getColumnCount(); c++) { //Looping throuhg rsmd to store the column names in the right
order.
                         ColumnNames[c] = rsmd.getColumnName(c+1);   //Store in ColumnNames to create table.
                         System.out.println(ColumnNames[c]);
                    }

                    int c = 0;
                    while (rs.next()) {

                         Table[c][0] = rs.getString(ColumnNames[0]);
                         Table[c][1] = rs.getString(ColumnNames[1]);
                         Table[c][2] = rs.getString(ColumnNames[2]);
                         Table[c][3] = rs.getString(ColumnNames[3]);
                         c++;
                    }

                      } catch (SQLException e){
                                        System.out.println("Error: " + e.getMessage());
                      }

               closeConn();

               //Create table
               TempTable = new JTable(Table, ColumnNames);
               TempTable.setVisible(true);
               TempTable.setCellSelectionEnabled(true);
               TempTable.setBackground(Color.pink);
               TempTable.getTableHeader().setReorderingAllowed(false);
               TempTable.setBounds(70, 50, 422, 226);

               }
}
```

LoadTable.java

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.JTable;

public class LoadTable extends LoadDataSava {

	private JFrame frame;
	static private JTable MYTABLE;
	static public String Data1[][];
	static public String [] ColumnNames1 = {"MesCyc_ID", "Mes_Press", "Mes_Diff", "Mes_Disch", "Mes_Vol"};


	/**
	 * Create the application.
	 */
	public LoadTable() {
		initialize();
	}

	/**
	 * Initialize the contents of the frame.
	 */
	private void initialize() {
		frame = new JFrame();
		frame.setBounds(100, 100, 549, 372);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.getContentPane().setLayout(null);

		Table(); //Initialize MYTABLE

		JButton Finish = new JButton("Finish and Menu");
		Finish.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				frame.dispose();
				LaunchWindow WL = new LaunchWindow();
			}
		});
		Finish.setBounds(205, 298, 137, 29);
		frame.getContentPane().add(Finish);

		if(Model.compareTo("0") == 0) {
			Model = "Conventional";
		} else {
			Model = "Novel";
		}

		JLabel Description = new JLabel(Date + " - " + Model + " - " + CycTime + " seconds" + " - Bot_ID: " + Bot_ID);
		Description.setBounds(80, 22, 368, 16);
		frame.getContentPane().add(Description);

		JScrollPane scrollPane = new JScrollPane(MYTABLE);
		scrollPane.setBounds(51, 64, 442, 222);
		frame.getContentPane().add(scrollPane);

		frame.setVisible(true);

	}
	/*
	 * This method opens the connection.     */
		private static void openConn() {
	    try {

	        // Try to connect to the database
	        conn = DriverManager.getConnection("jdbc:sqlite:" + FILE); //Need to have JDBC driver referenced.
	        MESSAGE = "DB \"" + FILE + "\" selected.";
	        System.out.println(MESSAGE);

	    } catch (SQLException EX) {
	        MESSAGE = "exception: " + EX;

	    }
	 }
```

```
            /*
            * This method closes the connection. It should be called by all methods who open a DB connection
            */
            private static void closeConn() {
                try {
                    if (conn != null) conn.close();

                } catch (SQLException EX) {
                    if ((EX.getErrorCode() == 50000) && ("XJ015".equals(EX.getSQLState()))) {
                        System.out.println("SQLite shut down properly");

                    } else {
                        System.err.println("SQLite did not shut down properly");
                        System.err.println(EX.getMessage());

                    }

                }

            }


           public static void Table() {

                      openConn();

                      try{
                       Statement stmt;
                       Statement stmt1;
                    ResultSet rs = null;
                    ResultSet rowNum = null;

                    String strSQL = "";

                    // Select all from the bottle with the Date in front
                    strSQL = "SELECT MesCyc_ID, Mes_Press, Mes_Diff, Mes_Disch, Mes_Vol "
                                + " FROM tMeasurements "
                                + " WHERE MesBot_ID = " + Bot_ID //Using the Bot_ID that was stored from the user parameter.
                                + " ORDER BY MesBot_ID ASC;";

                    stmt = conn.createStatement();
                    rs = stmt.executeQuery(strSQL); //ResultSet that stores executed script

                    //For the Number of rows for the 2D array
                    String strSQL1 = "SELECT COUNT(*) FROM tMeasurements WHERE MesBot_ID = " + Bot_ID;
                    stmt1 = conn.createStatement();
                    rowNum = stmt1.executeQuery(strSQL1);

                                //Get the number of rows to initialize array with right number of rows
                                Data1 = new String[(((Number) rowNum.getObject(1)).intValue())][5];


                    int c = 0; //Loop through the ResultSet in order to store the measurements in the 2D array to create the MYTABLE
                    while (rs.next()) { //While the resultSet has more items.

                         Data1[c][0] = rs.getString(ColumnNames1[0]);
                         Data1[c][1] = rs.getString(ColumnNames1[1]);
                         Data1[c][2] = rs.getString(ColumnNames1[2]);
                         Data1[c][3] = rs.getString(ColumnNames1[3]);
                         Data1[c][4] = rs.getString(ColumnNames1[4]);

                         c++;
                    }

                      } catch (SQLException e){
                                          System.out.println("Error: " + e.getMessage());
                      }



                      closeConn();

                      MYTABLE = new JTable(new MyTableModel(Data1, ColumnNames1));
                      MYTABLE.setCellSelectionEnabled(true);
                      MYTABLE.setBackground(Color.pink);
                      MYTABLE.getTableHeader().setReorderingAllowed(false);

                      ExcelAdapter myAd = new ExcelAdapter(MYTABLE);
//https://www.infoworld.com/article/2077579/java-tip-77--enable-copy-and-paste-functionality-between-swing-s-jMYTABLEs-and-excel.html
                      MYTABLE.setBounds(70, 50, 422, 226);

           }
}
```