Operating conduino through a MATLAB script

Novak Lab, Newcastle University

September 6, 2025

Conduino is a high-resolution multichannel water conductivity sensor. It uses micro-USB connectors for fast sensing of bipolar or tetrapolar liquid conductivity with minimal spatial granularity. The subsequent sections discuss a complete procedure for collecting data from the conduino board through a MATLAB script. Detailed information on conduino can be found here.

1 Getting the codes

Compiling and uploading Firmware Conduino Matlab.ino code is required to upload the arduino sketch to the conduino board. After uploading the arduino sketch to the board, the data can be collected from the controller board using the MATLAB script ConduinoMultichannel.m.

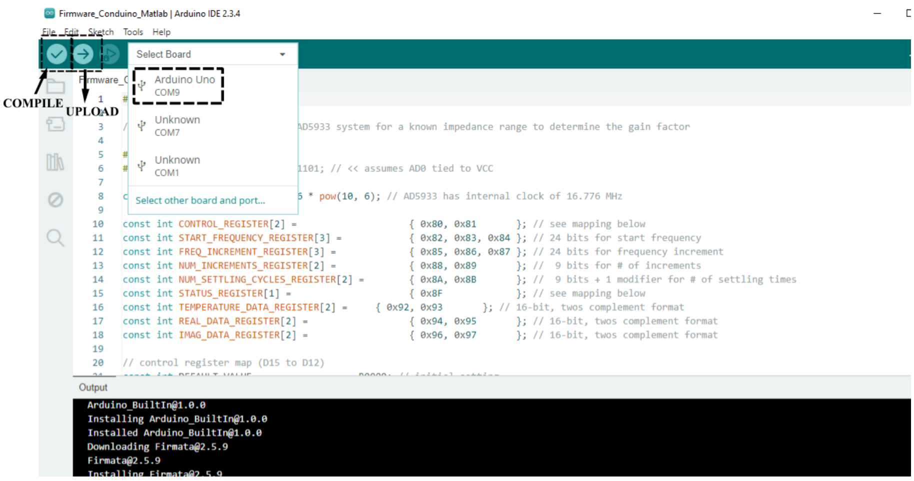


Figure 1: Arduino IDE interface

2 Uploading Arduino sketch on conduino board

An open source software Arduino IDE should be installed to upload arduino sketch (Firmware Conduino Matlab.ino) on the conduino board. An administration log-in is required for the software installation, compilation, and uploading of the code. After successful installation of the software, conduino board should be connected to the PC. A USB port powers the board, so there is no need to be a separate connection for the power supply. After connecting the board, open the (Firmware Conduino Matlab.ino code on Arduino IDE software. Figure 1 shows the page that will appear when you open the (Firmware Conduino Matlab.ino code. Select the *Arduino Uno* board, which will automatically appear after clicking the select board option. The serial port number will also appear with the *Arduino Uno* board (Figure 1). To compile the code, click the compile icon shown in Figure 1. After successful compilation, the code can be uploaded to the board by clicking the upload icon shown in Figure 1.

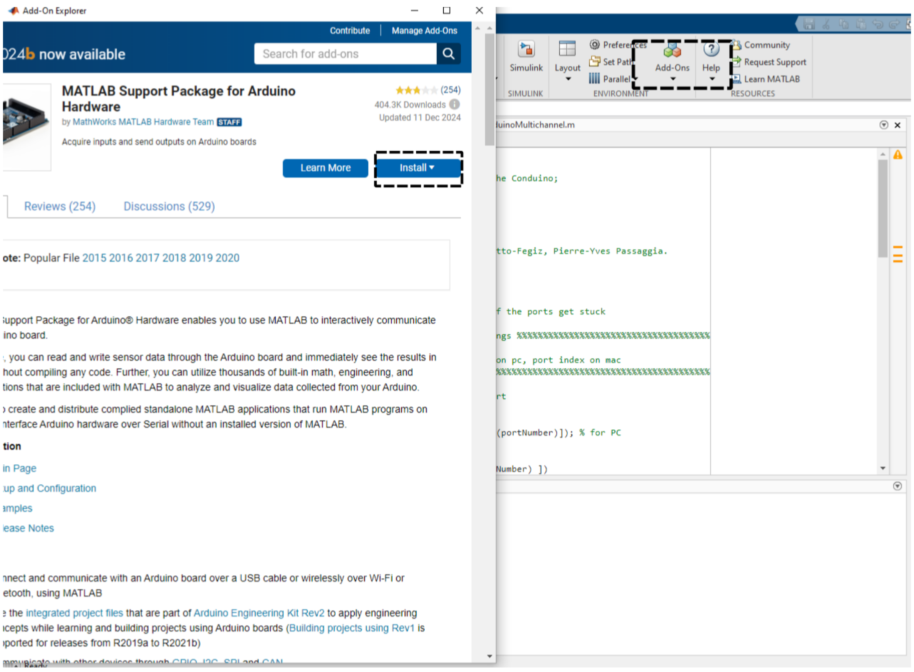


Figure 2: Step 1

3 Data collection through MATLAB script

To operate conduino through MATLAB, an add-on package *MATLAB Support Package for Arduino Hardware* should be installed from the *Add-Ons* tab of the *Environment* section from the MATLAB toolstrip (Figure 2). An administration log-in is required to complete the installation, and conduino board should be connected to the PC via USB port. After installing the package, a prompt will appear to proceed with the configuration. The configuration can be completed by following the instructions indicated in Figures 3, and 4. In Figure 4, the board and the port should be selected correctly; the correct board and port option will be prompted during selection. The libraries *I2C*, *SPI*, and *Servo* should be selected, and libraries are uploaded to the server by clicking the *program* tab. The *Test connection* tab should be clicked to check if all the steps are completed correctly (Figure 5). After a successful test, the configuration can be completed by clicking *Next*.

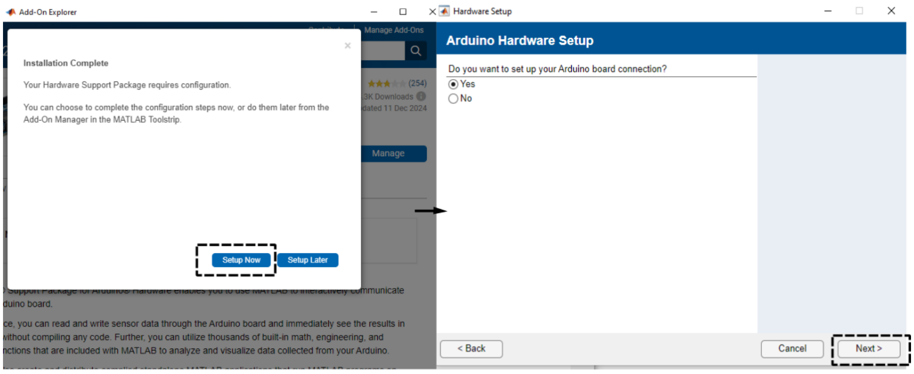


Figure 3: Step 2

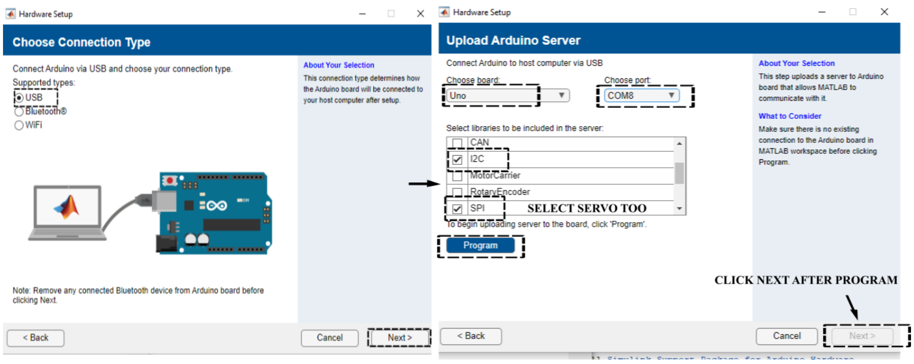


Figure 4: Step 3

After the successful completion of configuration, and with the board connected to the PC, open the ConduinoMultichannel.m script on MATLAB. Remember to change the port number in the code. The correct port number can be found by checking the COM port in the Arduino IDE interface shown in Figure 1. The data can be collected by running the script.

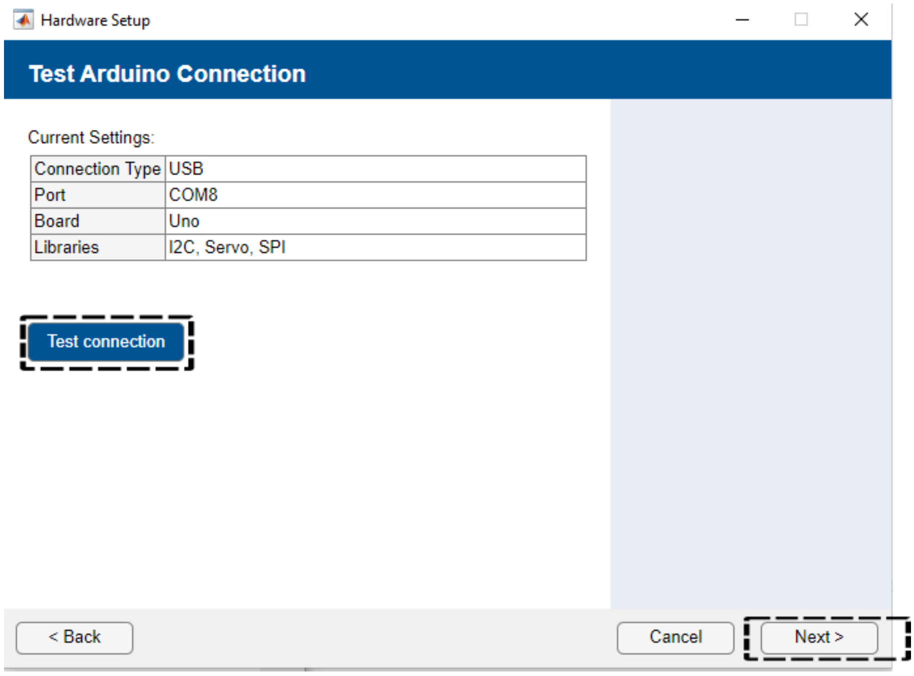


Figure 5: Step 4

4 Protocol for plotting background stratification

To obtain background stratification Stratification.m should be run. The code computes the density variation along the height moved by the rig. The steps to be followed to collect data from the probe are;

Step 1: Place the rig above the top layer of the water with its tip touching the water surface.

Step 2: Connect the Conduino Board.

Step 3: Select the appropriate port (refer to section 3).

Step 4: Run the code Stratification.m. In this step threshold value for the data to be filtered out should be determined from *Figure 1.fig*. The data below the data point corresponding to the circled portion in Fig. 6 should be filtered out.

Step 5: Run the code Stratification.m again after including the threshold data value in the code.

Step 6: Enter the number of samples (e.g. 300) and start the motor by pressing the green button on the stepper motor firmware.

Step 7: Save the Stratification.fig file generated after collecting data from probes in the folder containing the Stratification.m and Processed_Image_Tangent.m. Collect the data stored in array ch1[ ], C[ ], rho [ ], d [ ], and h[ ] and paste it in the Excel file (Probe Reading.xls) shown in Fig. 7.

Step 8: Run the code Processed_Image_Tangent.m to calculate the pycnocline thickness. The code generates a tangent on Stratification.fig and draws a horizontal line through the tangent point and calculates the distance between it.

Step 9: Save the *Sratification.fig*, *Raw Data.fig*, *Smooth_Stratified.fig* (if needed) and *Probe Reading.xls* in the folder *DDMMYY* that also contains digiflow data.

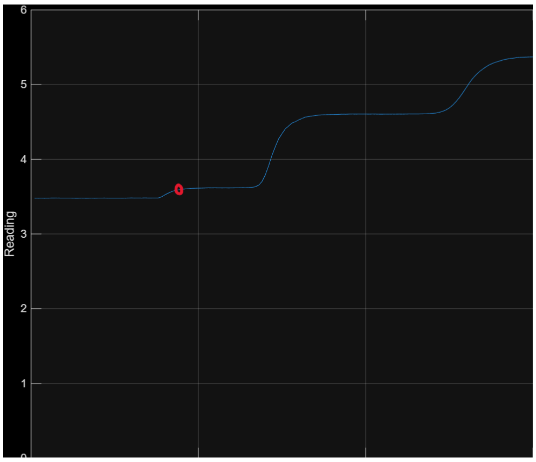


Figure 6: Threshold Data

Note: In step 1, placing the probe tip just at the surface of the top layer gives the start signal to filter out the data required from the raw data collected by the probe. As data only needed to be collected during the forward cycle of the rig movement, the initial rig placement also gives stop signal for collecting the required data. In step 5, running the code Stratification.m will collect data by the rig during the forward cycle only. The detailed explanation about the code is explained in the section 6. The code will generate three figures *Figure 1.fig*, *Figure 2.fig* and *Figure 3.fig*. Save the *Figure 1.fig* as *Raw Data.fig* (refer to step 9), *Figure 2.fig* as *Stratification.fig* (refer to step 7 and step 9), and *Figure 3.fig* as *Smooth_Stratified.fig* (refer to step 9). It is to be further noted that saving *Figure 3.fig* is only required when there is visible noise in the *Figure 2.fig*. Furthermore, in step 7, the *Stratification.fig* is stored in the folder with the code Processed_Image_Tangent.m for image processing. The generated image will give information about the pycnocline thickness

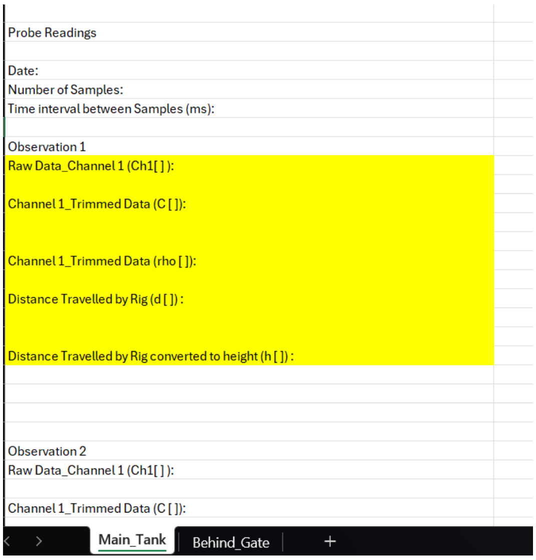
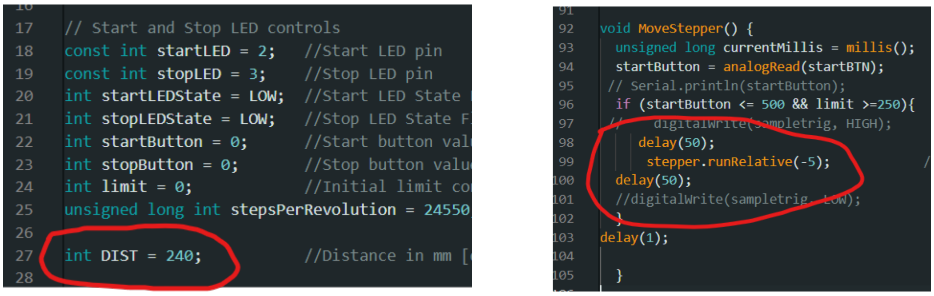


Figure 7: Probe Reading.xls

5 Code for changing the rig travel pattern

The distance moved by the rig can be modified by uploading code Stepper_Motor.ino and Stepper_motor_5mm_step.ino on the firmware that moves the stepper motor. To upload the code on the stepper motor firmware, connect the firmware to the computer via USB connection. It is to be noted that the rig should not be connected to other power source during the upload. The firmware derives power from the computer via USB connection; if the rig is connected to another power source, it will damage the firmware, as it will receive power from two sources. Open the code Stepper_Motor.ino on Arduino IDE interface to modify the rig travel distance. The total travel distance can be modified by changing line 27 of the code Stepper_Motor.ino highlighted in Fig. 8a. At present, it is configured to travel a distance of 240 mm (Fig. 8a). The rig movement can also be modified to travel in small steps by uploading the code on the stepper motor firmware. The step distance can be modified by modifying line 99 of the code as highlighted in Fig. 8b. The code now will move the rig in steps of 5 mm (Fig. 8b). It is to be noted that the firmware can be configured with only one code at a time.



(a) (b)

Figure 8: Stepper Motor Configuration

6 Matlab code for computation of background stratification

The code Stratification.m computes the probe data converted to the density reading along the vertical height of the tank. The theory behind the calculation of distance travelled by the rig is discussed in the section 8. In the present code, the rig travels 24 cms of total distance in the forward cycle, where the initial 1.2 cms it travels with acceleration given by equation 1. However, the data collected by probe during this period is filtered out, thus distance calculation during this period is not included in the code. The different portions of the code are explained in blocks as follows.

BLOCK 1 (Fig. 9) : The code takes input like port number and system type (mac or pc) from the user. It also defines the acceleration and velocity values determined by checking the time ti from the code (refer to section 8). In the block, different arrays are defined to store the time information from the probe in terms of filtered data. The array ch1[ ] stores all the raw data from the probe, while cht[ ] only stores data above threshold values. The threshold data is determined during step 4 of the section 4. Furthermore, chf[ ] stores the first half of the filtered out data from cht[ ]. The vartime [ ] stores the corresponding time of the cht[ ]. while recordtime [ ] will start the time counter from zero by taking readings from vartime [ ]. th [ ] stores the first half of the trimmed data from recordtime[ ] that also considers the initial 1.2 cms of rig movement in account, while thf [ ] will start the time counter from zero by taking readings from th [ ]. The array d [ ] and t [ ] is calculated according the information given in equations 1, 2 and 3. The C[ ] stores the final trimmed value of conductivity readings that is required to plot the stratification. rho [ ] stores the converted density values.

BLOCK 1

```
%stratification.m
clear
close all
delete(instrfindall); % Use this if the ports get stuck

%%%%%%%%%%%%%%%%%%%%%%%%%%%% Settings %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
macORpc = 'pc'; % 'mac' or 'pc'
portNumber = 10; % COM port number on pc, port index on mac
a1 = 10.41;%Motor acceleration and decceleration value
v1 = 5;%Motor Constant velocity
vartime = [];%array to store the time values corresponding to the conductivity reac
recordtime = []; %array that will start the time counter based on vartime
%d1 = [];%array to store the distance travelled by rig during accelaration
d2 = [];%array to store the distance travelled by rig with constant velocity
d3 = [];%array to store the distance travelled by rig during decceleration
d = [];%array to store the total distance travelled by rig
cht = [];%array to store the conductivity readings above threshold value
chf = [];%array to store the the first half of the conductivity readings above thre
th = [];%array to store the the time corresponding to the first half of the conduct
thf = [];%array that will start time counter based on th[]
%t1 = [];
t2 = [];%array to store the thf[] values when rig travels with constant velocityc
t3 = [];%array to store the thf[] when rig travels during deceleration
t = []; %array to store the t2[] and t3[] values
h = []; %array to store to convert the distance values from d[] to height starting
%htf = [];
C = [];%array that will store the values of the conductivity readings based on dist
rho = [];%array that will store the vdensity readings corresponding to the conducti
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 9: BLOCK 1

BLOCK 2 (Fig. 10): In Block 2 of the code, the code reads raw data from the serial port and stores the values in the arrays of the connected channels. In most cases, channel 1 is connected, and raw data is stored in ch[ ].

BLOCK 3 (Fig. 11): In Block 3 of the code, the data filtering is done, where the user gives the value of the threshold data below which all the data should be filtered out. The portion of the code also computes the first half of the probe data.

BLOCK 4 (Fig. 12): In Block 4 the distance computation are done. It is done by running two loops. each for the calculation of d2[ ] and d3[ ] respectively (equations 2, 3), The loop starts with t1 and ends at 9*t1 for d2 [ ], while for d3 [ ] it ends with 10*t1.

BLOCK 5 (Fig. 13): In Block 5 the conductivity reading is stored in array C [ ] for plotting against h [ ]. The h [ ] computation is also done during this block of the code. The conversion of conductivity readings from C [ ] to rho [ ] is also done in this block of the code. The conversion is done according to the equation 9.

BLOCK 6 (Fig. 14): Sometimes the obtained conductivity data has noise; in order to smooth the curve of the plot, data smoothing can be done if required. This is done in block 6 of the code. In this block, code to detect the point for drawing tangent through the curve is also added.

BLOCK 7 (Fig. 15): In this block, code to generate three figures discussed in step 8 of the section 4 is mentioned. It is to be noted a separate code Processed_Image_Tangent.m for drawing tangent and getting information about pycnocline thickness is also available. The tangent and thickness calculation by this code will appear as Fig. 16.

```
%assigns the object s to serial port
switch macORpc
    case 'pc'
        s1 = serial(['COM' num2str(portNumber)]); % for PC
    otherwise
        s = seriallist;
        disp(['Using port ' s(portNumber) ])
        s1 = serial(s(portNumber)); % for mac
end

set(s1, 'InputBufferSize', 128); %number of bytes in inout buffer
set(s1, 'BaudRate', 115200);
set(s1, 'Parity', 'none');
fopen(s1);
clc
prompt = 'How many samples? (dt ~40 ms typically) ';
nSamples=input(prompt,'s');
fprintf(s1,'%c',nSamples);
nSamples = str2double(nSamples);
ch1=zeros(0,nSamples);
ch2=zeros(0,nSamples);
ch3=zeros(0,nSamples);
ch4=zeros(0,nSamples);

tic
fwrite(s1, '%f');
for nSample=1:nSamples                                  BLOCK 2
    ch1(nSample)= fscanf(s1, '%f');
    ch2(nSample)= fscanf(s1, '%f');
    ch3(nSample)= fscanf(s1, '%f');
    ch4(nSample)= fscanf(s1, '%f');
```

Figure 10: BLOCK 2

```
for i = 1:length(time) %loop for storing conductivity readings above threshold value and the corresponding time
    if ch1(i) > 3.1 %threshold need to be confirmed from Figure1
        vartime = [vartime, time(i)];%it stores the succesive values of vartime
        recordtime = [vartime - vartime(1)];%it stores the succesive values of recordtime
        cht = [cht, ch1(i)];%it stores the succesive values of conductivity readings
    end                                                                          BLOCK 3

end
 halflength = floor(length(cht)/2); %code to store the first half of the conductivity readings and the corresponding vartime
chf = cht(1:halflength);
th = vartime(1:halflength);
for j = 1:length(th)
    %thf = [thf, th(j)-th(1) + 0.516];%code to store the successive values of thf
    thf = [thf, th(j)-th(1)];%code to store the successive values of thf
end
  ti = thf(end)/10;%the rig moves with deceleration for time ti.
```

Figure 11: BLOCK 3

7 Appendix

8 Rig travel distance calculation

The rig driven by a stepper motor travels with an initial 5 % of the distance with constant acceleration (a), next 90 % with constant velocity, and the last 5 % with constant deceleration. Thus, to calculate the distance (d) in time (t), let us consider the rig travels a distance of d_1 in time t_1 with acceleration, d_2 in time t_2 with constant velocity (v), d_3 in time t_3 with constant deceleration (a). Thus we will get,

```matlab
% the constant velocity
startTime = thf(1);%loop starts from initial value of thf
endTime = 9*ti;%loop ends at time when rig travelled with constant velocity
%maxIter = 13;
i = 1;% starting loop counter
while thf(i) >= startTime && thf(i) < endTime %loop will run during the constant velocity run of rig
        d2 = [d2, 1.2 + (thf(i)-thf(1))*v1];%distance calculation based on constant velocity, the threshold conductivity reading start
        t2 = [t2, thf(i)];
        i = i + 1;
end
% Portion of the code to calculate the distance travelled by rig during
% the deceleration of the motor

startTime = 9*ti;%loop starts from  value of thf when rig starts movement with constant deceleration
endTime = 10*ti;%loop ends when rig completes first half of the travel.
%maxIter = 13;
i = i(end);%loop starts counter right after the previous loop counter ends
while thf(i) >= startTime && thf(i) < endTime %loop during the deceleration run of rig
          d3 = [d3, d2(end) + (thf(i)-t2(end))*v1 - (thf(i)-t2(end))*(thf(i)-t2(end))*0.5*a1];
          t3 = [t3, thf(i)];% distance calculation during deceleration run of rig
        i = i + 1;
end
 d = [d2 d3];
```

BLOCK 4

Figure 12: BLOCK 4

```matlab
startTime = thf(1); %loop to filter the conductivity reading values based on distance calculation
endTime = t3(end);%the end time should be taken from the t3[] in order to remove any error arising due to data trimming
%maxIter = 13;
j = 1;
while thf(j) >= startTime && thf(j) <= endTime
          C = [C, chf(j)];
          h = [h, (41.2 - d(j))]; %it will convert the rig travel distnace to the height moved by rig in the tank starting from 40 cm
        j = j + 1;
end
t = [t2 t3]; % array to store time values from thf[] with size equal to C[];
startTime = thf(1); %loop to convert the conductivity reading to the corresponding density values
endTime = t3(end);%the end time should be taken from the t3[] in order to remove any error arising due to data trimming
%maxIter = 13;
k = 1;
while thf(k) >= startTime && thf(k) <= endTime
          rho = [rho, (1025 + ((C(k) - C(1))/(C(end) - C(1)))*25)];%linear interpolation of data for converting conductivity readings
        k = k + 1;
end
```

BLOCK 5

Figure 13: BLOCK 5

```matlab
rho_smooth = smoothdata(rho, 'sgolay', 15);
dh = gradient(h,rho_smooth);
d2h = gradient(dh, rho_smooth);
% it will detect the index of the point where tangent to be drawn
idx_inflect = find(diff(sign(d2h))); % it will detect the index of the point where tangent to be drawn

%dh = gradient(h,rho);
%d2h = gradient(dh, rho);
% Choose the index of the point where you want the tangent
%idx_inflect = find(diff(sign(d2h))); % Change this index as needed (between 2 and length(x)-1)



plot(ch1);% Plots the total number of samples (figure1)
hold on;
plot(ch2);
plot(ch3);
plot(ch4);
grid on
xlabel('Sample')
ylabel('Reading')
```

BLOCK 6

Figure 14: BLOCK 6

$$d_1 = 0.5at_1^2 \quad \rightarrow \quad 0.05d = 0.5at_1^2 \quad \rightarrow \quad a = \frac{0.1d}{t_1^2} \quad and \quad v = at_1 \quad \rightarrow \quad v = \frac{0.1d}{t_1} \tag{1}$$

$$d_2 = vt_2 \quad \rightarrow \quad 0.9d = \frac{0.1d}{t_1}t_2 \quad \rightarrow \quad 9t_1 = t_2 \tag{2}$$

$$d_3 = vt_3 - 0.5at_3^2 \quad \rightarrow \quad 0.05d = \frac{0.1d}{t_1}t_3 - 0.5\frac{0.1d}{t_1^2}t_3^2 \quad \rightarrow 1 - 2\frac{t_3}{t_1} + \frac{t_3^2}{t_1^2} = 0 \quad \rightarrow t_3 = t_1 \tag{3}$$

Thus, it can be found out that the rig travels a total distance d in total time $11t_1$. The total rig travel distance can be set by using the codes given in section 5. The time t_1 value can be found by dividing the thf[end] (Stratification.m) by 11. In the code Stratification.m, the t_1 is calculated as ti, which is calculated by dividing thf[end] by 10. As in the code Stratification.m the initial acceleration period is not considered this is why the thf[end] value is divided by 10.

```
plot(rho_smooth,h);%plots the variation of smoothed density with the height
hold on;
%plot(time,ch2);
%plot(time,ch3);
%plot(time,ch4);
grid on
for n = 1:length(idx_inflect)%plots the tangent in the same figure
    i0 = idx_inflect(n);
    rho0 = rho_smooth(i0);
    h0 = h(i0);
    slope = dh(i0);

    tan_rho = linspace(rho0 - 0.5, rho0 + 0.5, 100);
    tan_h = h0 + slope*(tan_rho - rho0);
    %plot(rho0, h0);
    %plot(rho0, h0, 'ro', 'MarkerSize', 8, 'LineWidth', 2);
    %plot(tan_rho, tan_h, 'r--', 'Linewidth', 1.5);
    plot(tan_rho, tan_h);
    %plot([min(rho) max(rho)], [h0 h0]); %plots and horizontal line through
    %the tangent point.

end
xlabel('Density (kg/m^3)')
ylabel('Height')                                    BLOCK 7
%legend('ch1', 'ch2', 'ch3', 'ch4')
pp=axis;
axis([1000 1060 10 50])

```

Figure 15: BLOCK 7

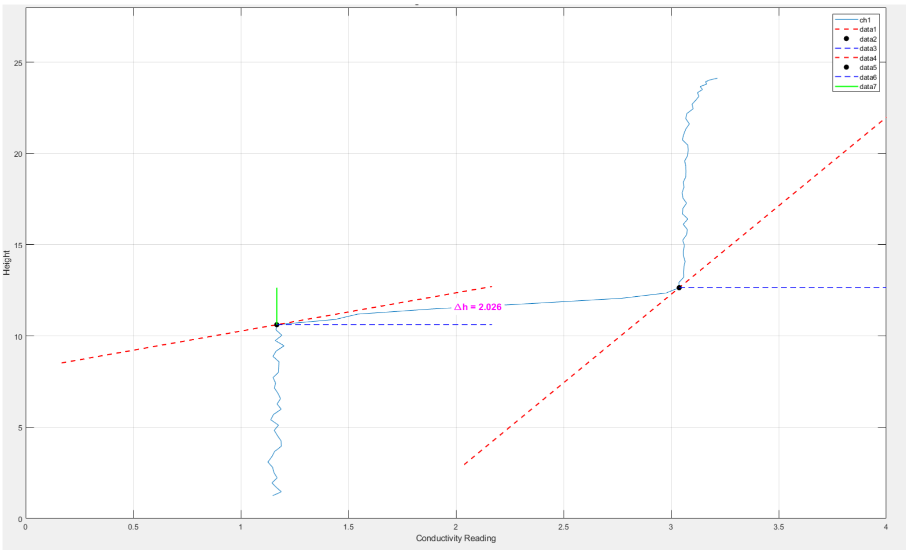


Figure 16: Code for Image Processing

9 Linear interpolation formula

The density information corresponding (values in the array rho[ ]) to the conductivity readings (values in the array C[ ]) can be obtained from linear interpolation. The corresponding density values are obtained by the given relation,

$$\rho_i = \rho_0 + \frac{(\rho_n - \rho_0)}{(C_n - C_0)}(C_i - C_0) \tag{4}$$

where ρ_0, ρ_i, and ρ_n are the density of the upper layer, the i^{th} layer and the lower layer. C_0, C_i and C_n are the first element, i^{th} element, and last element of C[ ].