A Study and Implementation of the Control System of Flap Type Water Wave Maker

By

Yue Zheng

Thesis
Submitted to Flinders University
for the degree of

Master of Engineering (Electrical and Electronic)

College of Science and Engineering
23-05-2024

TABLE OF CONTENTS

ABSTRACT

Previous water wave study usually depended on water wave generating equipment or computational simulation. Generating water wave in water flume was a common method in this field. Several research have verified and validated the water wave theory and the applicable condition for piston type and flap type wave paddle model. A laboratory-sized water wave maker prototype has been designed and been utilised in Flinders University. It could generate water wave by a wave paddle controlled by a stepper motor. This project implemented a water wave maker control system via Arduino developing board and Arduino developing platform to fulfill precise control to the stepper motor. The system could configure the water wave parameters via a simplified user interface in either wireless mode or stand-alone mode. Meanwhile, the project also studied two relevant water wave equations and applied them to the control system for verification in real system. The control system was functionally good and could be upgraded in the future. The verification was partially satisfactory. The applicable conditions of the wave equations in this water wave maker and control system have been found. Some calibrations and compensations could be done to improve this system. Overall, the thesis provided a user perspective for thinking and conducting a project, and collected experiment data to verify the theory and find its limitations.

DECLARATION

I certify that this thesis:

1. does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university
2. and the research within will not be submitted for any other future degree or diploma without the permission of Flinders University; and
3. to the best of my knowledge and belief, does not contain any material previously published or written by another person except where due reference is made in the text.

Signature of student...

Print name of student...

Date...

I certify that I have read this thesis. In my opinion it is/is not (please circle) fully adequate, in scope and in quality, as a thesis for the degree of <Degree Name>. Furthermore, I confirm that I have provided feedback on this thesis and the student has implemented it minimally/partially/fully (please circle).

Signature of Principal Supervisor...

Print name of Principal Supervisor...

Date...

ACKNOWLEDGEMENTS

I’d like to acknowledge the following for their help in this project:

- Prof. Youhong Tang for mentoring and supervising this project.

- Yunzhong (Steven) Wang for assisting, instructing, and co-supervising throughout the entire project.

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Background

Over 71% of the planet is made up of the ocean, which offers a wide range of substantial and plentiful resources, such as food, minerals, energy, etc. As a result, human study of ocean waves is becoming increasingly significant. Humans can simulate and analyse ocean waves with the use of wave generators. Although certain wave generators have been constructed, their size and cost are usually their main drawbacks.

The water wave maker is a useful instrument for examining how different wave amplitudes and frequencies affect ocean wave performance. The simulator may also be used to test similarly scaled ships or floating items. It may also be used as a teaching tool to illustrate different wave situations.

Yunzhong (Steven) Wang, a PhD candidate at Flinders University, is now working on a laboratory-scale wave maker prototype. The wave maker generates water wave using a transmission belt and stepper motor. The majority of the wave simulator system's parts were made with 3D printing, which significantly lowers material waste during the development phase.

Scope and Hypotheses

There are two constraints on the prototype. Firstly, it is unable to control wave height and frequency at the same time. Secondly, it should also include a remote control.

Due to the preceding section's information, this project will concentrate on the relationships between the motor's rotational speed and steps, the wave paddle movement and frequency, and the water wavelength and wave height. Moreover, figure out a way to provide wireless control and design a user interface.

Objectives and Methodology

The goal of this project is to test whether the water wave theory is valid in real water wave generating system. Meanwhile, programming will be implemented on Arduino platform.

LITERATURE REVIEW

Although there are many solutions available for water wave generating today, they may be roughly divided into two categories. A wave maker can be emulated by a numerical implementation, or by an implementation that is physically similar to a wave generator.

There are two main types of water wave maker, piston type and flap type.

Piston type

Goring's approach (Goring, 1978) is used to express the average horizontal velocity of the water. Four generation rules were investigated both analytically and empirically in the work by (Katell & Eric, 2010) in contrast with Goring's technique. Two generation laws were derived using the Lagrangian formulation in the first-order (Korteweg & De Vries, 1895) and second-order shallow water theories (Temperville, 1985), while the other two laws were produced using Rayleigh's solitary wave solution.

One limitation that should be mentioned is that because the paddle movements were constrained by jack length, the maximum dimensionless amplitude produced by the Rayleigh solution would be less than those obtained. The experimental findings demonstrated that each of the four techniques could produce a water wave that was suitable. However, the waves produced by the Rayleigh solution were easier to build and more desired.

(Wu, et al., 2016) used both Goring's approach and the modified Goring's method that (Wu, et al., 2014) offered to confirm the stability of the generated solitary waves. They employed a more complex method to calculate the wave speed C, boundary outskirt decay coefficient K, and free surface displacement $\eta(x)$ from the (Fenton, 1972) study. The outcomes of the experiment demonstrated that, in comparison to Goring's method, the modified approach could produce more stable water waves, particularly at higher wave heights. They included an explanation of the wave height discount as well. This phenomenon is said to have been generated by water leaking between the wave paddle and the water tank on both the sides and the bottom.

Flap type

(Lal & Elangovan, 2008) used the Computational Fluid Dynamics (CFD) approach to evaluate the correctness of the wave maker theory (WMT). Their research revealed that:

(a) the CFD method agrees with WMT theory; (b) the wave height H and wavelength λ are proportional to paddle stroke S; (c) the wave height H and wavelength λ increase as water depth h increases; and (d) the wave height H and wavelength λ are proportional to paddle time T with a fixed flap to time period ratio. Additionally, they discovered that a flap's tiny time period and angle might not generate a clear wave. They came to the conclusion that, in order to generate regular waves, CFD modelling can replace wave makers.

(Eldeen, et al., 2021), drawing on earlier research, reported a more precise ratio of the maximal paddle stroke S and wavelength λ to be 1/20 in shallow water and 1/25 in deep water. Additionally, they made a significant move into irregular waves, which were likewise supported by experimental evidence.

There are two main limitations on the forementioned WMT. Initially, it is only applicable within a certain range of wave steepness, H/λ (Krvavica, et al., 2018). Second, before the wave steepness H/λ was reached, the measured values are slightly smaller than the values predicted, according to the experimental results. (Krvavica, et al., 2018) proposed a generalised equation for both regular and broken waves. The wave number k is actually controlled by the wave dispersion relation (Dean & Dalrymple, 1991). Water leaking around the paddle (Madsen, 1970), which also happened in (Wu, et al., 2016)'s experiment, was the reason for the little variation.

There is a water leakage issue surrounding the paddle of both types of wave makers, which lowers the water wave height.

METHODOLOGY

Overview

Structure of the Wave Maker

The water wave maker was designed and assembled by Steven Wang. Figure 1 is the overview drawing of the system. Figure 2 is the system with water flume in the laboratory at Tonsley Campus, Flinders University.

As shown in Figure 1, component 1 is a water flume, component 2 is a dual-track placed on the flume for mounting other components, component 3 is a transmission belt connected and driven by the motor module (component 4) and supporting module (component 7), component 5 is the paddle driver with teeth for connecting the transmission belt and a ball bearing for driving the wave paddle (component 6), component 8 is used to fix the components on the track.

The core part of the motor module (component 4) is PD60-4-1076 including a stepper motor and driver electronics, which is manufactured by Trinamic, German. The motor has up to 256 micro steps per full step resolution so that it offers precise movement control.

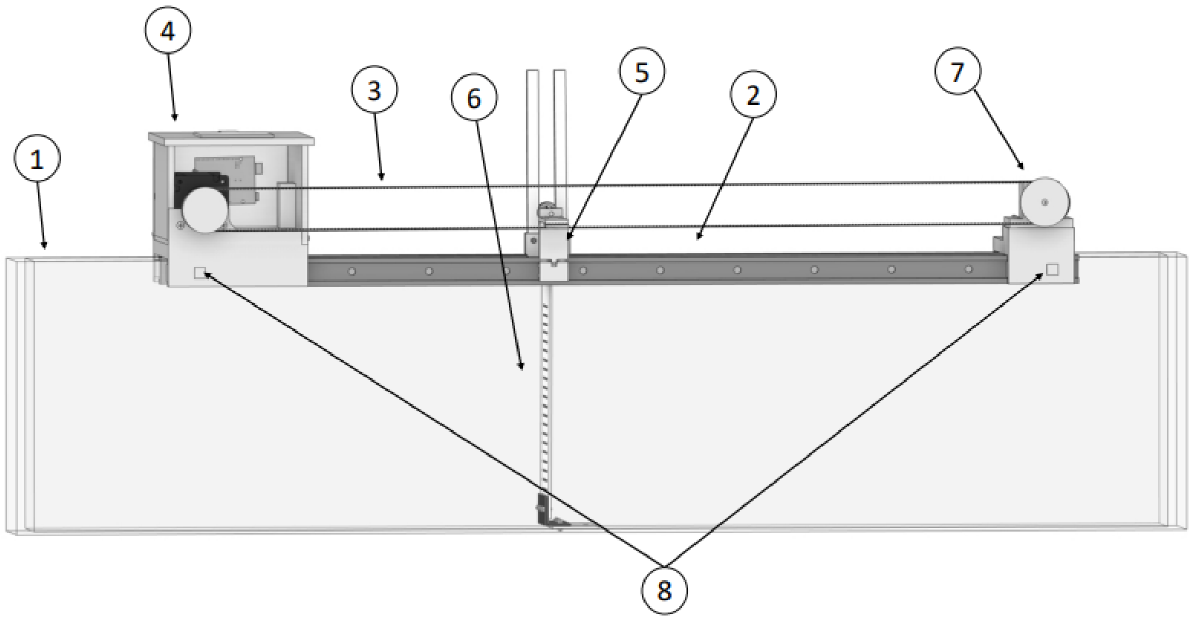


Figure 1. Overview drawing of the system

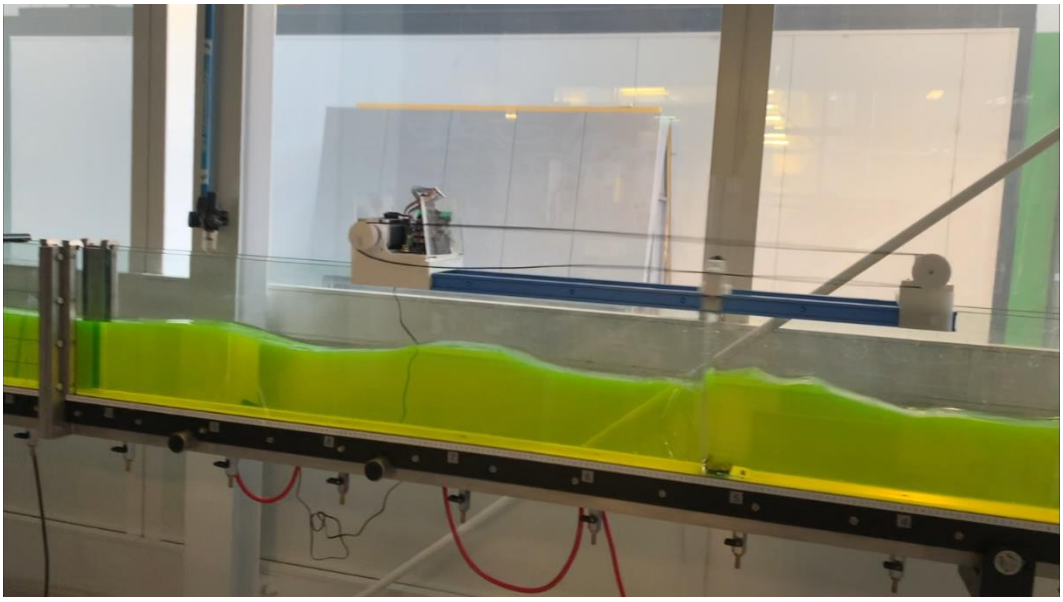


Figure 2. The system with water flume in the laboratory

Study Perspective

As the wave maker actually generates water wave by the wave paddle controlled by the stepper motor, there are multiple variables that can be controlled. For the stepper motor, the steps and the rotation direction are two main variables. For the wave paddle, the flap angle and the frequency can easily be controlled. For the generated water wave, wavelength and wave height are most intuitive to observers.

Considering that the system is designed for water wave study, and it is mostly used by those who expertise in water wave study, the user perspective ran throughout the entire project. In other words, not only the previous mentioned variables were studied, but also a user interface was designed for setting and modifying the water wavelength and wave height. Other variables about paddle and motor were all invisible to user.

Mathematical Derivation

Wave Equation to Paddle Movement

According to the equation proposed by (Dean & Dalrymple, 1991), the ratio of wave height H to the paddle stroke S is

$$\frac{H}{S} = \frac{4\sinh(kh)}{kh} \cdot \frac{kh\sinh(kh) - \cosh(kh) + 1}{\sinh(2kh) + 2kh} \tag{1}$$

where h is the still water depth and k is the wave number given by

$$k = \frac{2\pi}{\lambda} \tag{2}$$

where λ is the wavelength.

In practical scenario, the still water depth h is a constant. Water wavelength λ and wave height H are required at a specific value. With these two values, wave number k can be obtained using (2) and then paddle stroke S is

$$\begin{aligned} S &= \frac{H}{\frac{4\sinh(kh)}{kh} \cdot \frac{kh\sinh(kh) - \cosh(kh) + 1}{\sinh(2kh) + 2kh}} \\ &= H \cdot \frac{kh}{4\sinh(kh)} \cdot \frac{\sinh(2kh) + 2kh}{kh\sinh(kh) - \cosh(kh) + 1} \end{aligned} \tag{3}$$

As shown in Figure 3, $S = h \cdot \tan\theta$, where θ is the flap angle.

On the other hand, (Dean & Dalrymple, 1991) also proposed water wave dispersion relation

$$\omega^2 = gk\tanh(kh) \tag{4}$$

where $\omega = 2\pi f$ is the angular frequency, f if paddle frequency, and g is gravitational acceleration. Thus, paddle frequency

$$f = \frac{\sqrt{gk\tanh(kh)}}{2\pi} \tag{5}$$

Therefore, with the required water wavelength and wave height at a fixed still water depth h, wave number k is calculated and then the paddle stroke S and paddle frequency f are determined using (3) and (5).

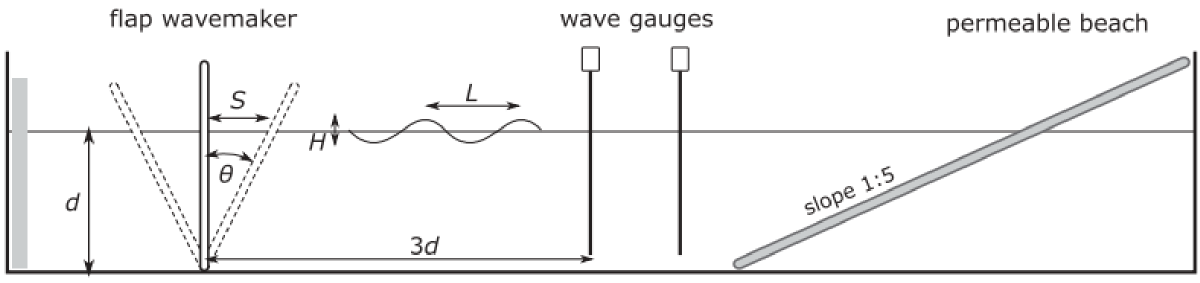


Figure 3. Sketch of the water wave maker

Paddle Movement to Motor Control

The paddle stroke S is the maximum flap distance at the surface of the water. As the paddle is driven by the ball bearing above the water, the transmission belt must move longer (see Figure 4). So

$$\tan\theta = \frac{S}{h} = \frac{D}{M} \tag{6}$$

where D is the transmission belt moving distance, M is the ball bearing height. Hence

$$D = \frac{M}{h}\cdot S \tag{7}$$

Equation (7) indicates the distance the transmission belt should move for driving the wave paddle flapping at paddle stroke S at a certain frequency. This can be converted to motor steps.

In this system, the stepper motor rotates 200 full steps per revolution, the transmission belt will move accordingly by 14.4 cm. If the belt moves D, the number of motor steps

$$n = \frac{200}{14.4}\cdot D = \frac{200}{14.4}\cdot\frac{M}{h}\cdot S \tag{8}$$

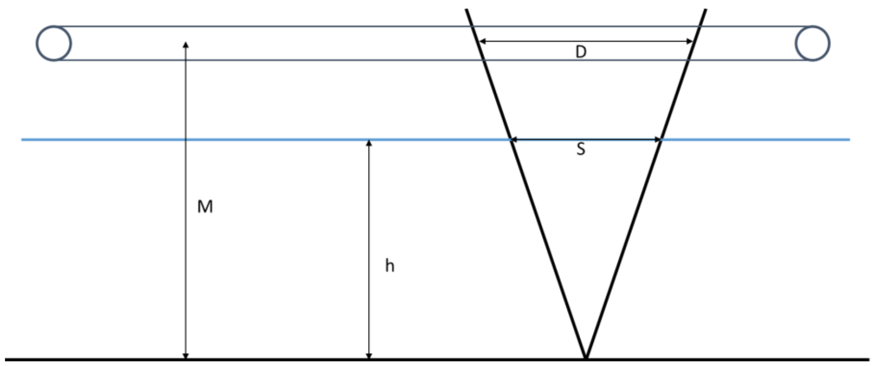


Figure 4. Simplified sketch of the flap paddle and transmission belt

Besides, the frequency at which the paddle flap is also the frequency that the motor changes its rotation direction. Obviously, in every time period T, the wave paddle flaps forth and back once, i.e., the motor should move n full steps in half time period.

The stepper motor is driven by switching high and low level of voltage. In this study, Pulse Width Modification (PWM) wave, which is simpler to manipulate, was used to drive the stepper motor. And the stepper motor was in default setting, 16 micro steps per full step. As shown in Figure 5, the PWM wave had a 50% duty cycle, which led to equal pulse width for high and low voltage level. Each pulse generated one micro step, which meant

$$2 \cdot \text{pulseWidth} \cdot 16 \cdot n = \frac{T}{2}$$

$$\text{pulseWidth} = \frac{T}{64n} \tag{9}$$

Substitute (8) into (9),

$$\text{pulseWidth} = \frac{T}{64n} = \frac{14.4}{64 \cdot 200} \cdot \frac{h}{M} \cdot \frac{T}{S} = \frac{14.4}{64 \cdot 200} \cdot \frac{h}{M} \cdot \frac{1}{S \cdot f} \tag{10}$$

Now, the motor steps could be properly controlled through STEP PIN on the motor module. And the rotation direction was switched by setting high and low voltage level through DIR PIN on the motor module.

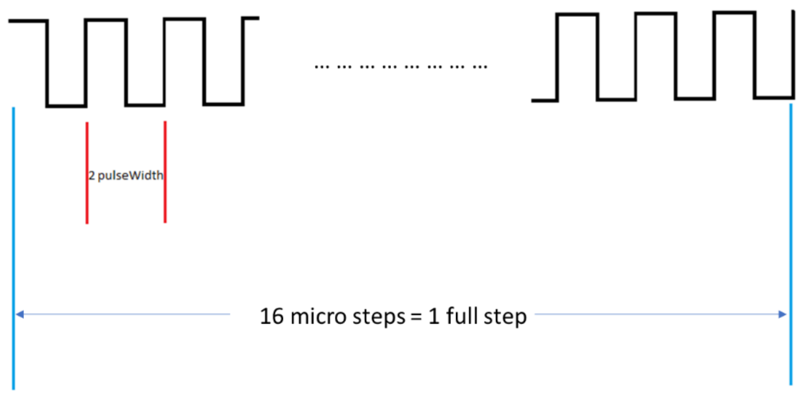


Figure 5. PWM signal and motor steps

Figure 6 shows the chain of variables of the system. Once the user set the required water wavelength and wave height, the system will do the rest. This was explained in the next sections.

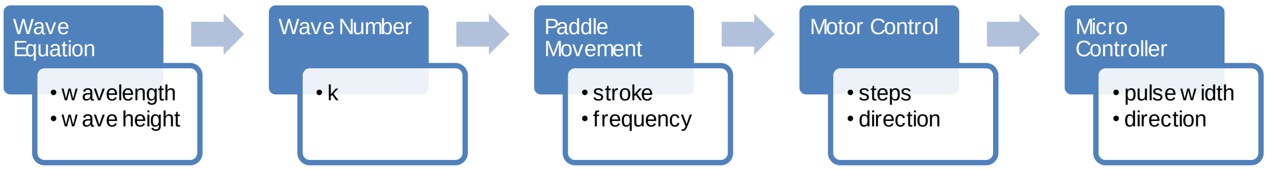


Figure 6. Chain of variables

Programming

The programming was implemented on the Arduino platform. The developing board was Arduino UNO R3, and the developing environment was Arduino IDE 2.3.3. The connection is shown in Figure 7.

This stage was split into two phases: Keypad control and Remote control.

Keypad control was used throughout the developing phase, and it could be set to stand-alone system for future expansion if necessary.

Remote control was realised by Bluetooth technology. It has high communication speed and low energy consumption, which are suitable for laboratory using.

There were several general settings that applied for both control mode: PIN definition, Water flume parameters, Water wave parameters, and the mathematical equations (i.e., equations (3), (5), (9)). All of these were to be initialised when the system was turned on.

The full code is attached in appendices.

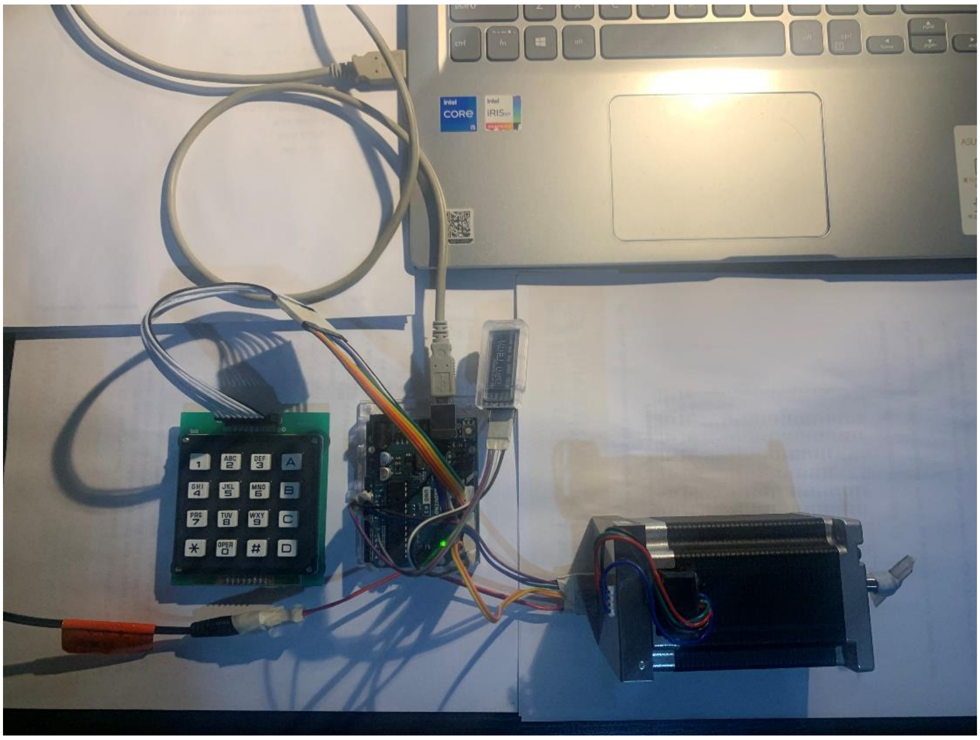


Figure 7. Connection of all components

Keypad Control

The Keypad was a 4×4 number pad. Figure 8 shows the diagram of the connection. Figure 9 shows the code for configuration. Note that the external library Keypad.h must be included in the format: `#include <Keypad.h>` in order to use the class and subroutines.

As the keypad had only 16 press keys, there were 9 pre-set wave parameters switched by keys ‘1’ to ‘9’ initially. Key ‘0’ was defined as ‘default’. Key ‘#’ was defined as ‘start’. The motor was limited to run 5 time periods for testing and debugging the code.

With the progress of the programming, keys ‘1’ to ‘9’ were no longer used. Instead, keys ‘A’ and ‘B’ were used to increase and decrease water wavelength, keys ‘C’ and ‘D’ were used to increase and decrease wave height. Key ‘0’ was defined to display detailed settings.

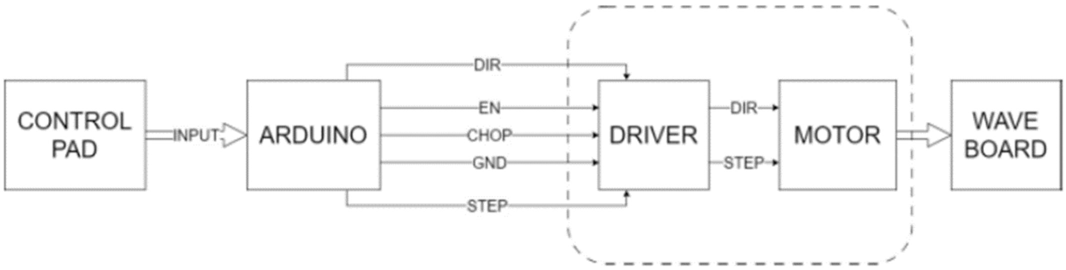


Figure 8. Diagram of the connection

```
/* Constants for row and column sizes */
const byte ROWS = 4;
const byte COLS = 4;

/* Array to represent keys on keypad */
char hexaKeys[ROWS][COLS] = {
  { '1', '2', '3', 'A' },
  { '4', '5', '6', 'B' },
  { '7', '8', '9', 'C' },
  { '*', '0', '#', 'D' }
};

/* Connections to Arduino */
byte rowPins[ROWS] = { 4, 5, 6, 7 };
byte colPins[COLS] = { 8, 9, 10, 11 };

/* Create keypad object */
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

/* keypad input */
char padInput;
```

Figure 9. Code for keypad configuration

Remote Control

Initially, this was serial communication through COM port on PC via Arduino Serial Monitor. The data was transmitted or received digit by digit between Arduino board and PC. There were two problems when data was sent to Arduino board. The first problem is: the previous sending might not be finalised, or the serial buffer was not empty. This would lead to unexpected inputs. To solve this problem, a subroutine ClearBuffer() was created to check and clear the serial buffer before new data was sent. The second problem is: how to receive every digit of the data and store it into one variable. This problem was addressed by subroutine GetSerial(). When the Serial Monitor set to “New Line” (see Figure 10), every message sent is ended by ‘\n’ that indicates a new line. Thus in the subroutine, the receiving message was terminated when the serial read was ‘\n’. Figure 11 shows subroutines ClearBuffer() and GetSerial().

Bluetooth communication was realised by a HC-05 Bluetooth Module. The sketch is shown in Figure 12. In order to activate the Bluetooth communication, the module should pair with the PC first, and then the communication port should be changed.

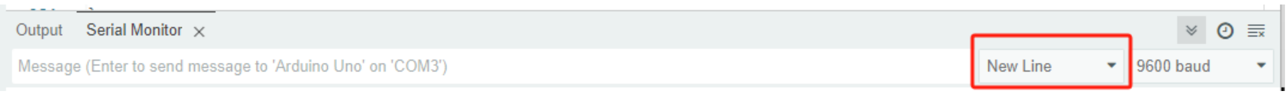


Figure 10. Screenshot of the Serial Monitor

```
void ClearBuffer(void) {  // clear serial buffer to avoid unexpected input
  while (Serial.available() > 0) {
    Serial.read();
  }
}

void GetSerial(void) {  // store serial input into an array, only for setting l & H
  static byte j = 0;
  char endMarker = '\n';  // mark the end of input. Serial monitor: New line
  char msg;               // receive every digit from serial input

  while (Serial.available() > 0 && inputReceived == false) {
    msg = Serial.read();
    if (msg != endMarker) {
      serialInput[j] = msg;
      j++;
      if (j >= num) {
        j = num - 1;
      }
    } else {
      serialInput[j] = '\0';  // terminate the string
      j = 0;
      inputReceived = true;
    }
  }
}
```

Figure 11. Code for ClearBuffer() and GetSerial()

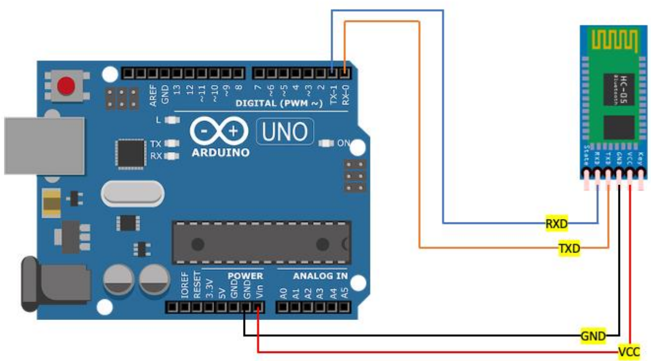


Figure 12. Sketch of Bluetooth connection

Motor Setting

The stepper motor was driven by two subroutines: Rotate(dir) and RunMotor(). The Rotate(dir) subroutine simply let the motor rotate required steps clockwise or counterclockwise at a corresponding frequency. The RunMotor() subroutine made the motor rotate back and forth periodically only when the motor is ready to run. It could be stopped by sending ‘a’ through serial port or pressing ‘*’ via keypad. The code is shown in Figure 13. Note that the three variables I, t1 and t2 are only for recording the time period, and it was explained in next section.

```
void RunMotor(void) {  // run motor back and forth
  if (motorReady == true) {
    int i = 0;
    long t1 = micros();  // record start time
    while (Serial.read() != 'a' && customKeypad.getKey() != '*') {
      Rotate(HIGH);  // motor rotates clockwise, flap paddle moves left
      delayMicroseconds(pulseWidth);
      Rotate(LOW);  // motor rotates counter-clockwise, flap paddle moves right
      delayMicroseconds(pulseWidth);
      i++;
    }
    long t2 = micros();  // record end time
    Serial.print(F("Running T = "));
    Serial.print((t2 - t1) / i);  // print execute time
    Serial.println(F(" us"));
    Serial.println();
  } else Serial.println(F("Motor not ready!"));
}

void Rotate(int dir) {                  // motor rotates at a required direction
  if (dir) PORTD |= B00000100;          // if dirPin set to HIGH
  else if (!dir) PORTD &= ~B00000100;   // if dirPin set to LOW

  for (int i = 0; i < drivingSteps; i++) {
    PORTD |= B00001000;              // stepPin set to HIGH
    delayMicroseconds(pulseWidth);   // time delay for HIGH
    PORTD &= ~B00001000;             // stepPin set to LOW
    delayMicroseconds(pulseWidth);   // time delay for LOW
  }
}
```

Figure 13. Code for Rotate(dir) and RunMotor()

User Interface

Interactive Logic Based on Serial Monitor

In order to get user inputs, the interactive dialog was needed. Based on the feature of Serial Monitor, a simplified user interface was designed. The interactive logic is shown in Figure 14. In the Home Menu, the user can select to enter either Pad control mode or Serial control mode. In the middle level, the user can display details on the screen, change water flume settings, change water wave settings, run the stepper motor, switch to the other control mode, and return to Home Menu. The bottom level consists of all the subroutines that do the corresponding operation to the system.

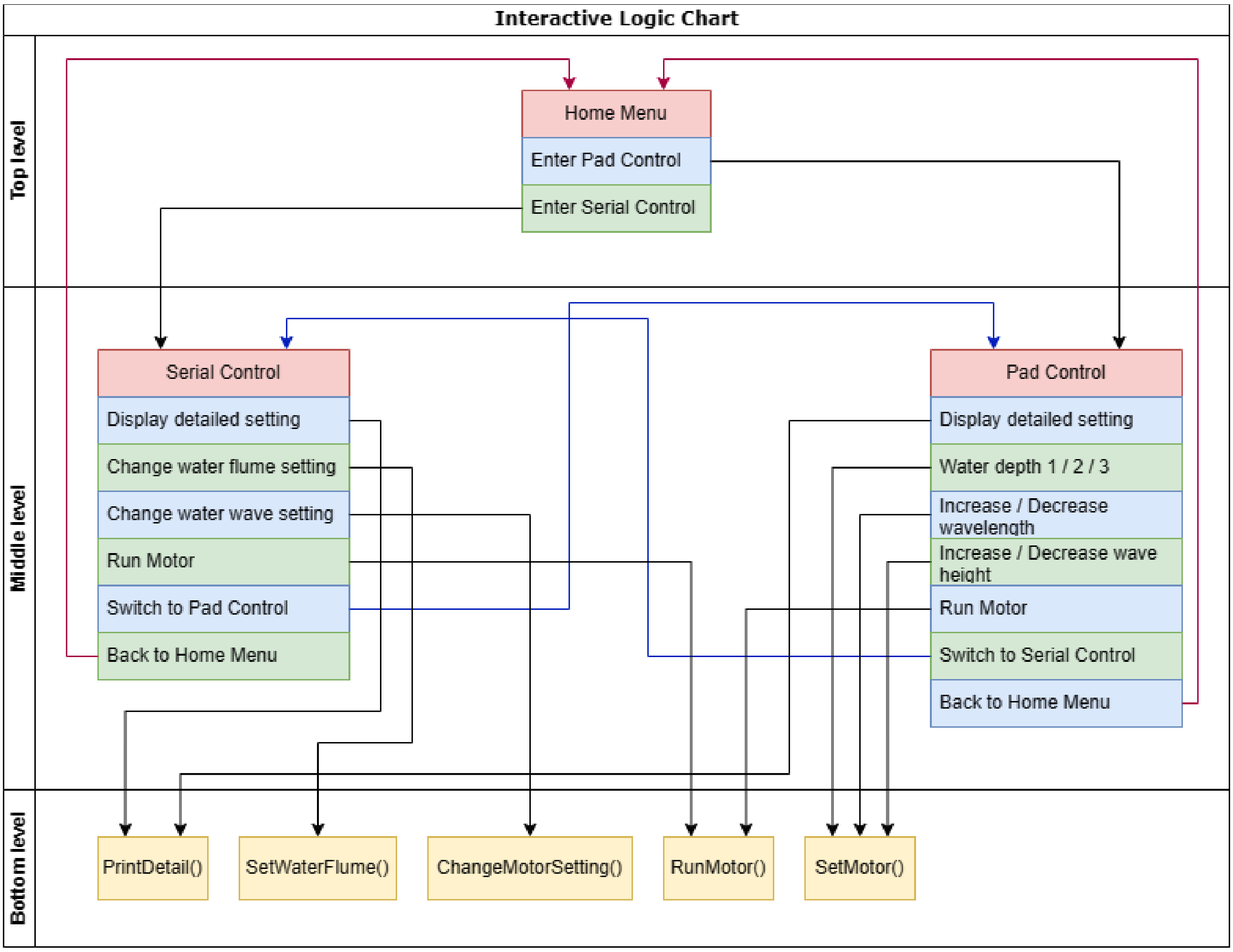


Figure 14. interactive logic of the user interface

Test and Debug

Debug and First Test

There were various bugs encountered at this stage, most of which were tiny and easy to fix. Still, there were some vicious bugs that led to system failure. Table 1 shows the bug list, cause, and solution.

Table 1. Bug list and solutions

Bug list	Cause	Solution
pulseWidth was always 0 second	'float' has only two decimal places but pulseWidth is microsecond scale	multiply pulseWidth by 1000000 and change the unit to microsecond
wavelength or wave height was set to invalid value	the value was assigned without checking after conversion	define a temporary variable to store the value, check if it is within valid range, then assign it to wavelength or wave height, otherwise pop up error message

the motor could not change direction smoothly, especially at high speed	the inertia of the rotor	add time delay between each direction change to release the kinetic energy
invalid or unexpected input	serial buffer was not empty	clear buffer every time before new inputs
cannot return to Home menu but switch to the other control mode	the switch between two control mode caused misdirection	add a variable 'switchMark' to indicate the correct operation. -1: Serial mode to Pad mode; 1: Pad mode to Serial mode; 0: normal back;

The first test was a 'water free' test, just to make sure the stepper motor correctly did what it asked to. An important question was whether the practical frequency was consistent with the theoretical frequency. An Arduino function micros() was used to measure the flapping period. The micros() function can record the time in microsecond since the board began running current program. Additionally, a counter was needed to record how many times the motor rotated back and forth. As mentioned before (see Figure 13), t1 recorded motor start time, t2 recorded motor stop time, and i recorded loop times. The practical time period was T = (t2 - t1) / i.

The measurements were shown in Table 2. Obviously, the errors in the light orange column. were unacceptable. The reason for the error was the executing time of each line of code. Initially, digitalWrite() was used to set the voltage level of STEP PIN. Actually, it did not happen instantly but took a few microseconds. Direct pin control can extremely reduce the executing time. The time periods were measured again after the code was modified (see Figure 13), and the errors in light blue column were negligible.

Table 2. Comparison between digitalWrite() and PIN control

λ (cm)	H (cm)	h (cm)	M (cm)	T (μs)	actual time period (μs) digitalWrite()	error%	actual time period (μs) PIN control	error%
40	2	15	35	510866.72	535978	4.92%	515839	0.97%
40	4	15	35	510866.72	554090	8.46%	513810	0.58%
40	6	15	35	510866.72	575841	12.72%	515422	0.89%
60	2	15	35	647517.70	680644	5.12%	652448	0.76%
60	4	15	35	647517.70	706072	9.04%	649682	0.33%
60	6	15	35	647517.70	731590	12.98%	647004	-0.08%
80	2	15	35	787495.13	830634	5.48%	792917	0.69%
80	4	15	35	787495.13	864706	9.80%	789273	0.23%
100	2	15	35	933010.72	989592	6.06%	941988	0.96%
100	4	15	35	933010.72	1028200	10.20%	932629	-0.04%
120	2	15	35	1083050.71	1151719	6.34%	1093863	1.00%

120	4	15	35	1083050.71	1207869	11.52%	1091792	0.81%

Second Test

The second test was 'on-water' test for the first time. Figure 2 shows how the system was set up.

The aim was to test whether the motor module worked normally, and to find the limitations and constraints of the real system. There were three factors that affected: the size of the water flume limited the water wave height, the length of the dual-track limited the water wavelength, and the maximum motor speed (i.e., the minimum pulseWidth) limited the flapping frequency. Table 3 shows the working range of the water wave maker.

Table 3. Working range of the water wave maker

Variable	Working range
water wavelength (cm)	$40 \leq \lambda \leq 120$
water wave height (cm)	$H \leq 7$
flapping frequency (Hz)	$f < 2$

Third Test

The aim of third test was to verify the mathematical equations. The still water depth $h = 15\ \mathrm{cm}$, the ball bearing height $M = 35\ \mathrm{cm}$. The results were taken by a high-speed camera. The pictures were attached in Appendices. Table 4 shows the comparison between the required wave parameters and measured wave parameters read from the pictures.

Final Test

From the third test, the generated water wave had required wavelength. But the wave height was not consistent with the required value. These outcomes indicated the correct frequency and insufficient flap stroke. The way of the paddle flapping was change in final test. The results were shown in Table 5.

RESULTS AND DISSCUSION

Third test results

As shown in Table 4, the real water wave generated by the paddle had required wavelength λ. But the some of the wave heights were less than required value, especially when the wavelength was small. The reasons were: (a) the transmission belt was not tight enough; (b) the water leaked from the gap between the wave paddle and the water flume, which was also stated by previous study and literature; (c) small wavelength required high frequency, resulted that the paddle flapped at a small amplitude.

Table 4. Comparison between the required wave parameters and measured wave parameters

Required				Measured	
λ (cm)	H (cm)	h (cm)	M (cm)	λ (cm)	H (cm)
40	3	15	35	40	1.5
40	6	15	35	40	4
60	3	15	35	60	3
60	6	15	35	55	4
80	3	15	35	80	3
100	3	15	35	100	3

Final test results

Table 5 shows the measured water wavelength and wave height at three different still water depths. Figure 15 shows the line charts respectively. Similar to third test, it clearly showed that the measured wavelength λ was in agreement with the required value.

When the still water depth was in between 10 cm and 15 cm, the matching range of the water wavelength was from 80 cm to 120 cm. If the wavelength was less than 60, the wave height would be less than it required. The reason was stated in the previous section.

When the still water depth was up to 20 cm, the wavelength still met the required, but the wave height was at least 30% less than the required value. A possible reason was that deeper water caused more load to the motor, resulting in incomplete rotation of the motor, which led to the fact that the paddle stroke was less than that it should be.

Table 5. The measured water wavelength and wave height at three different still water depths

Required					Measured			
λ (cm)	H (cm)	h (cm)	M (cm)	T (μs)	λ (cm)	H (cm)	actual time period (μs) PIN control	error%

40	2	10	35	528695.98	40	1.5	529741	0.20%
60	2	10	35	701865.66	65	1.5	704390	0.36%
80	2	10	35	884307.21	80	2	896466	1.37%
100	2	10	35	1072917.93	95	2	1075476	0.24%
120	2	10	35	1265363.54	120	2	1277022	0.92%
40	2	15	35	510866.72	40	1	515839	0.97%
40	4	15	35	510866.72	40	3	513810	0.58%
40	6	15	35	510866.72	40	4	515422	0.89%
60	2	15	35	647517.70	60	2	652448	0.76%
60	4	15	35	647517.70	60	4	649682	0.33%
60	6	15	35	647517.70	60	5	647004	-0.08%
80	2	15	35	787495.13	80	2	792917	0.69%
80	4	15	35	787495.13	80	4	789273	0.23%
100	2	15	35	933010.72	100	2	941988	0.96%
100	4	15	35	933010.72	100	4	932629	-0.04%
120	2	15	35	1083050.71	120	2.5	1093863	1.00%
120	4	15	35	1083050.71	120	4	1091792	0.81%
40	4	20	35	507236.37	40	2	512042	0.95%
40	6	20	35	507236.37	40	3	510768	0.70%
60	4	20	35	629568.27	60	2.5	634962	0.86%
60	6	20	35	629568.27	60	4	630452	0.14%
80	4	20	35	747689.03	80	3	751910	0.56%
80	6	20	35	747689.03	80	4	752046	0.58%
100	4	20	35	868296.71	100	2	876448	0.94%
100	6	20	35	868296.71	100	4	873320	0.58%
120	4	20	35	992587.94	120	3	997379	0.48%
120	6	20	35	992587.94	120	3.5	990574	-0.20%

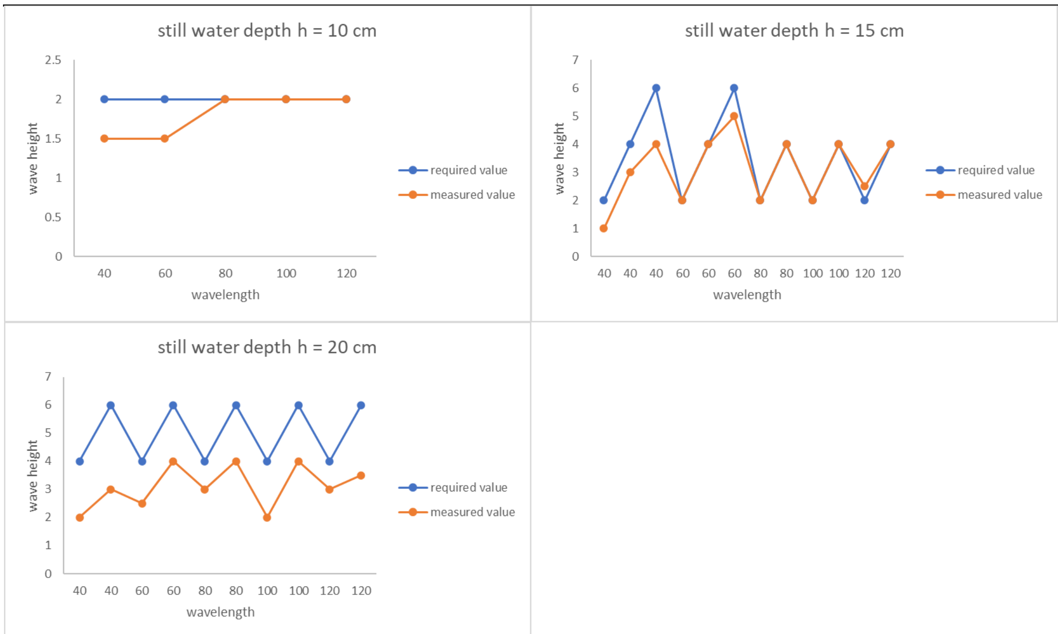


Figure 15. The measured water wavelength and wave height at three different still water depths

CONCLUSIONS

The water wave maker was designed by Yunzhong (Steven) Wang. Based on this prototype, the study and implementation of the control system was able to proceed. This project focused on two aspects: verify the water wave equations in real system rather than computer simulation, and design a control system prototype including user interface, mode select, and remote control.

Wave Equations

In the experiments, the wave equation was applicable for getting water wavelength from 40 cm to 120 cm. When the still water depth was in between 10 cm and 15 cm, the water wave height was valid if the water wavelength was in between 80 cm and 120 cm. The water wave equation was invalid when still water depth was up to 20 cm.

Possible reasons for the invalidation were: (a) loose of transmission belt, (b) water leakage around the wave paddle, (c) incomplete movement of the wave paddle at high frequency, (d) larger load or strong damping because of water depth.

Control System Prototype

The outcomes of this part were: (a) a dialog-like user interface was implemented, (b) the system could work in either Pad control mode or Serial control mode, (c) remote communication was realised by Bluetooth technology, (d) both water flume and water wave settings could be changed by user, (e) other system settings could be configured by modifying the source code, (f) the system was modular programmed for future modifications and expansions.

FUTURE WORK

All the problems stated in previous sections indicate the future work. (a) The wave equation needs compensation at small wavelengths and deep water. (b) Add sensor and actuator to realise close loop control. (c) Design graphic user interface.

BIBLIOGRAPHY

Boussinesq, J., 1871. Théorie de l'intumescence liquide appelée onde solitaire ou de translation se propageant dans un canal rectangulaire,. *CR Acad. Sci. Paris,* Volume 72, pp. 755-759.

Dean, R. G. & Dalrymple, R. A., 1991. *Water wave mechanics for engineers and scientists.* s.l.:world scientific publishing company.

Eldeen, A. S. S., El-Baz, A. M. & Elmarhomy, A. M., 2021. CFD modeling of regular and irregular waves generated by flap type wave maker. *Journal of Advanced Research in Fluid Mechanics and Thermal Sciences,* 85(2), pp. 128-144.

Fenton, J., 1972. A ninth-order solution for the solitary wave. *Journal of fluid mechanics,* 53(2), pp. 257-271.

Goring, D. G., 1978. *Tsunamis - The Propagation of Long Waves onto a Shelf,* Pasadena, California: California Institute of Technology.

Katell, G. & Eric, B., 2010. Accuracy of solitary wave generation by a piston wave maker.. *Journal of Hydraulic Research,* pp. 321-331.

Keating, T., Webber, N. & Havelock, T., 1977. The Generation of Periodic Waves in a Laboratory Channel: a Comparison between Theory and Experiment. *Proceedings of the Institution of Civil Engineers,* 63(4), pp. 819-832.

Korteweg, D. J. & De Vries, G., 1895. On the change of form of long waves advancing in a rectangular canal, and on a new type of long stationary waves. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science,* 39(240), pp. 422-443.

Krvavica, N., Ružić, I. & Ožanić, N., 2018. New approach to flap-type wavemaker equation with wave breaking limit. *Coastal Engineering Journal,* 60(1), pp. 69-78.

Lal, A. & Elangovan, M., 2008. CFD simulation and validation of flap type wave-maker. *International Journal of Mathematical and Computational Sciences,* 2(10), pp. 708-714.

Madsen, O. S., 1970. Waves generated by a piston-type wavemaker. In: *Coastal Engineering.* s.l.:s.n., pp. 589-607.

Temperville, A., 1985. Contribution à l’étude des ondes de gravité en eau peu profonde. *These d’Etat, Université Joseph Fourier—Grenoble I 0.66,* 1(2.83), p. 1.72.

Wu, N.-J., Hsiao, S.-C., Chen, H.-H. & Yang, R.-Y., 2016. The study on solitary waves generated by a piston-type wave maker. *Ocean Engineering,* Volume 117, pp. 114-129.

Wu, N.-J., Tsay, T.-K. & Chen, Y.-Y., 2014. Generation of stable solitary waves by a piston-type wave maker. *Wave Motion,* 51(2), pp. 240-255.

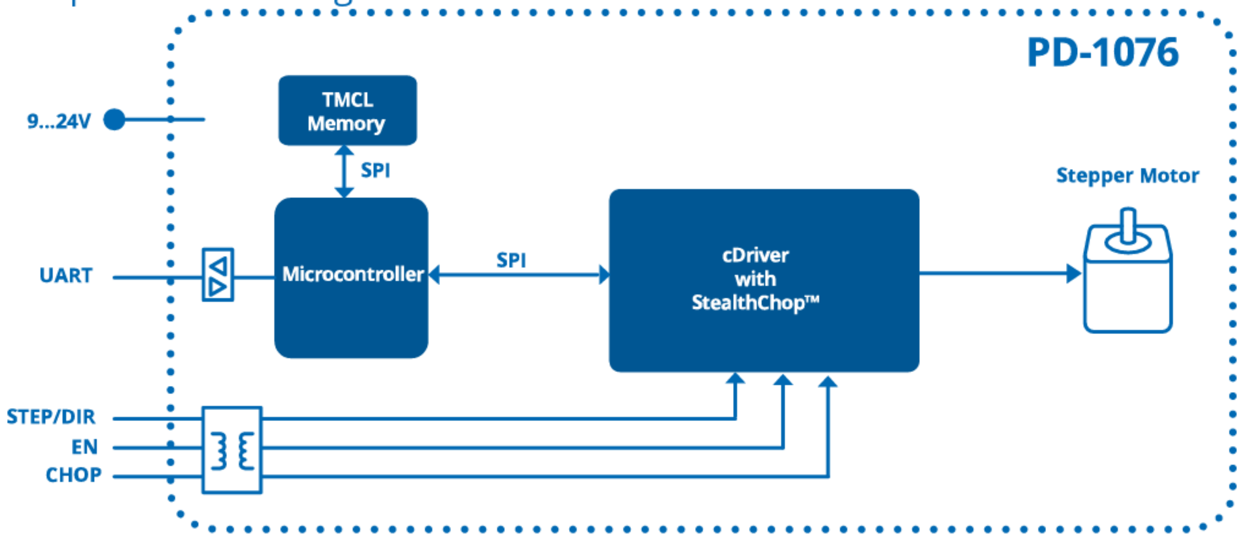
APPENDICES

Stepper Motor Features and Block Diagram

Features

- Supply Voltage +10…+30V DC
- Up to 3A RMS motor current
- Step and direction interface
- MicroPlyer™ to 256 microsteps
- StealthChop™ silent PWM mode
- SpreadCycle™ smart mixed decay
- StallGuard2™ load detection
- CoolStep™ autom. current scaling
- UART configuration interface

Simplified Block Diagram



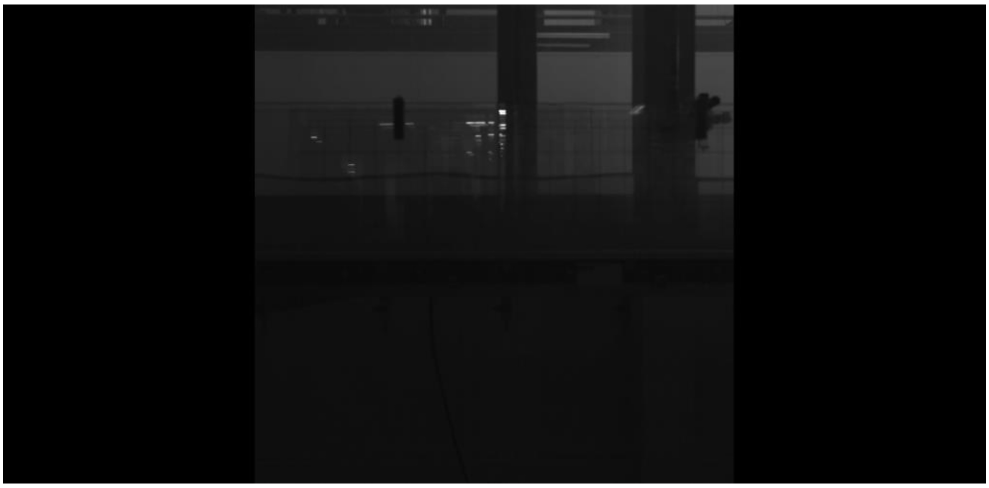
Arduino Board Tech Specs

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

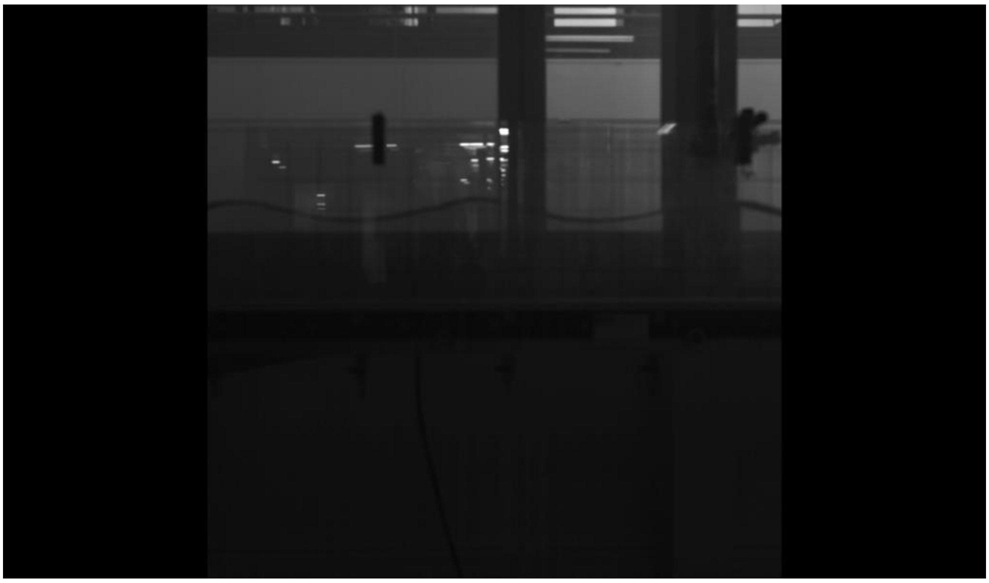
Bluetooth Module Features

- Sensitivity (Bit error rate) can reach -80dBm,The change range of output's power: -4 - +6dBm.
- Has an EDR module; and the change range of modulation depth: 2Mbps - 3Mbps.
- Has a build-in 2.4GHz antenna; user needn't test antenna.
- Has the external 8Mbit FLASH
- Can work at the low voltage (3.1V~4.2V). The current in pairing is in the range of 30～40mA.
- PIO control can be switched.
- This module can be used in the SMD.
- It's made through RoHS process.
- The board PIN is half hole size.
- Has a 2.4GHz digital wireless transceiver.
- Bases at CSR BC04 Bluetooth technology.
- Has the function of adaptive frequency hopping.
- Small (27mm×13mm×2mm)
- Peripherals circuit is simple.
- It's at the Bluetooth class 2 power level.
- Storage temperature range: -40 ℃ - 85℃， work temperature range: -25 ℃ - +75℃
- Any wave inter Interference: 2.4MHz， the power of emitting: 3 dBm.
- Bit error rate: 0. Only the signal decays at the transmission link, bit error may be produced. For example, when RS232 or TTL is being processed, some signals may decay.

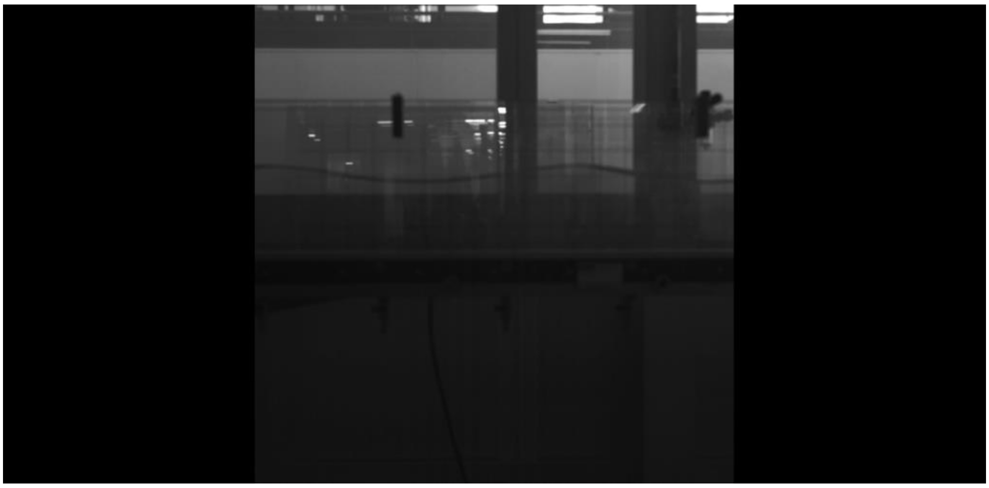
Third Test Screenshots



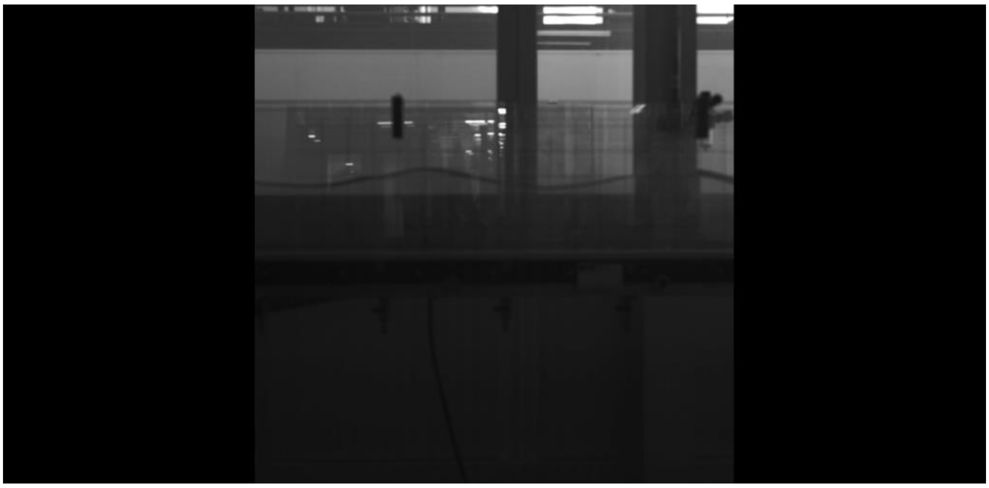
Still water depth = 15 cm, wavelength = 40 cm, wave height = 3 cm



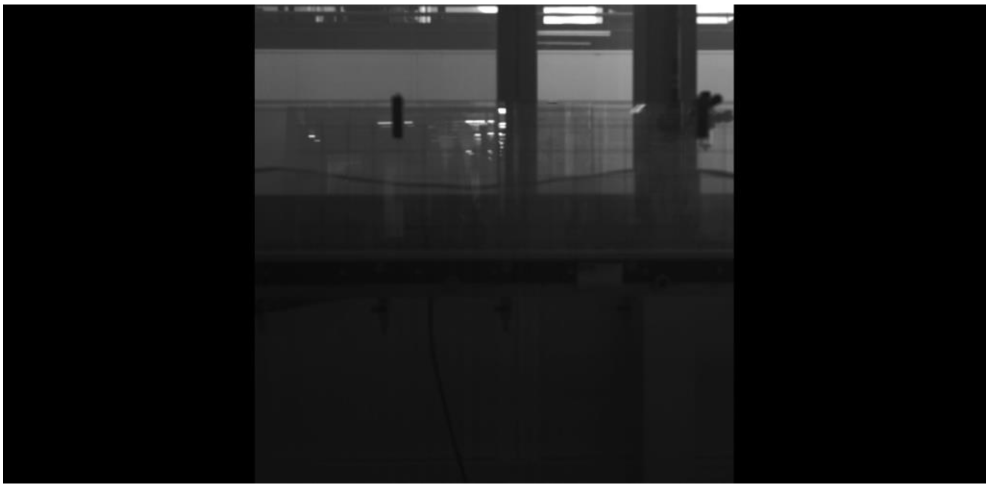
Still water depth = 15 cm, wavelength = 40 cm, wave height = 6 cm



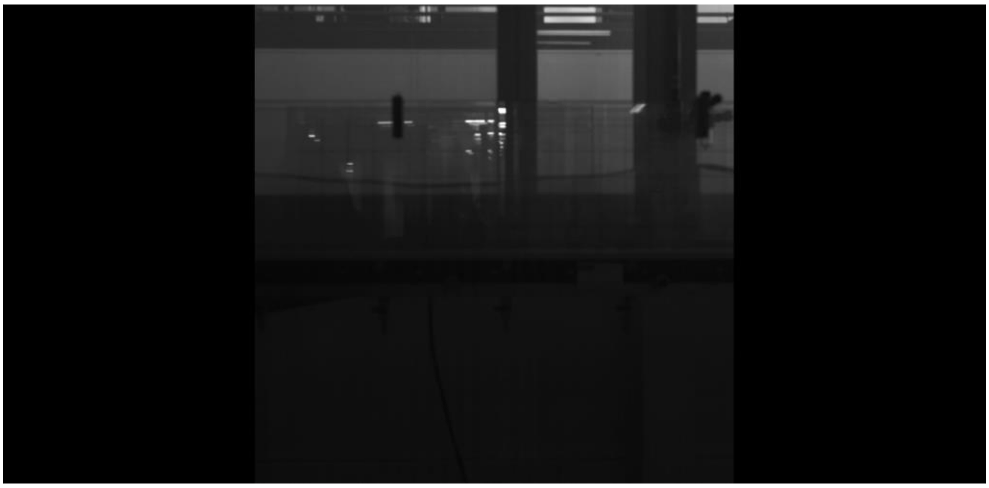
Still water depth = 15 cm, wavelength = 60 cm, wave height = 3 cm



Still water depth = 15 cm, wavelength = 60 cm, wave height = 6 cm

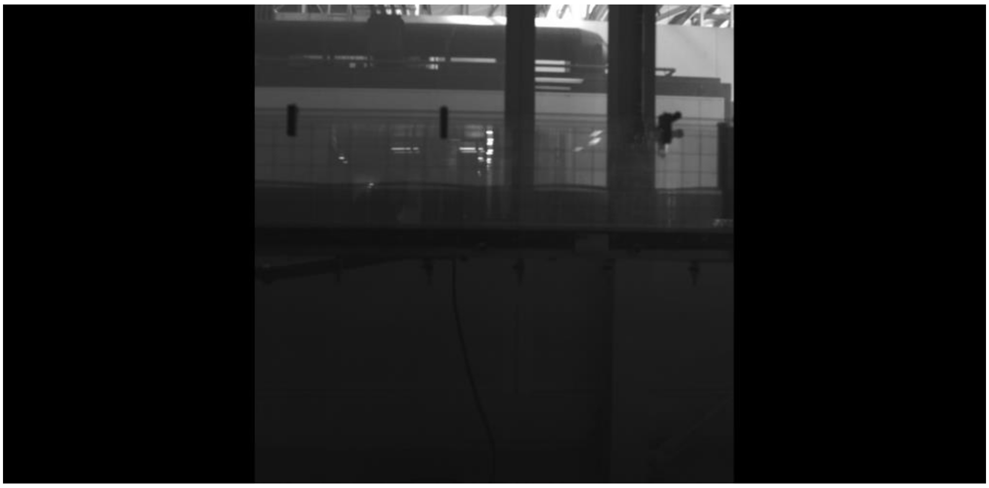


Still water depth = 15 cm, wavelength = 80 cm, wave height = 3 cm

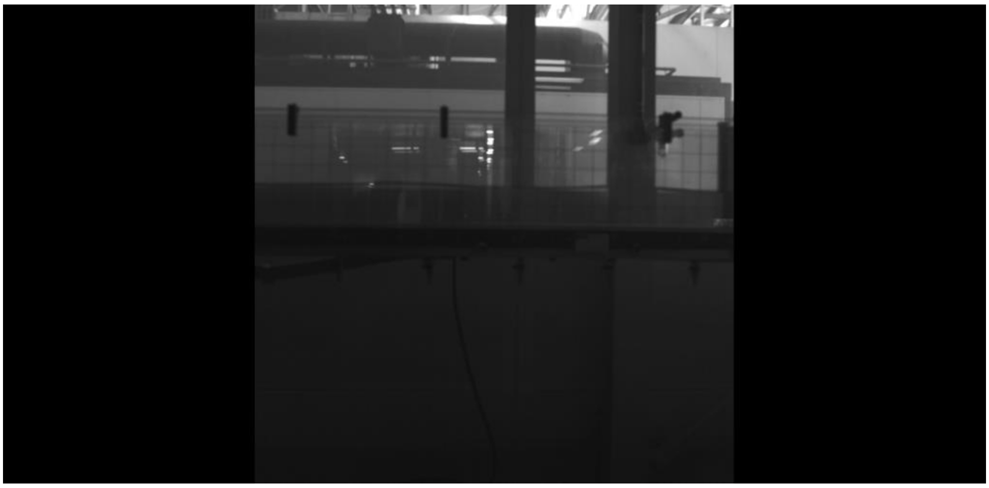


Still water depth = 15 cm, wavelength = 100 cm, wave height = 3 cm

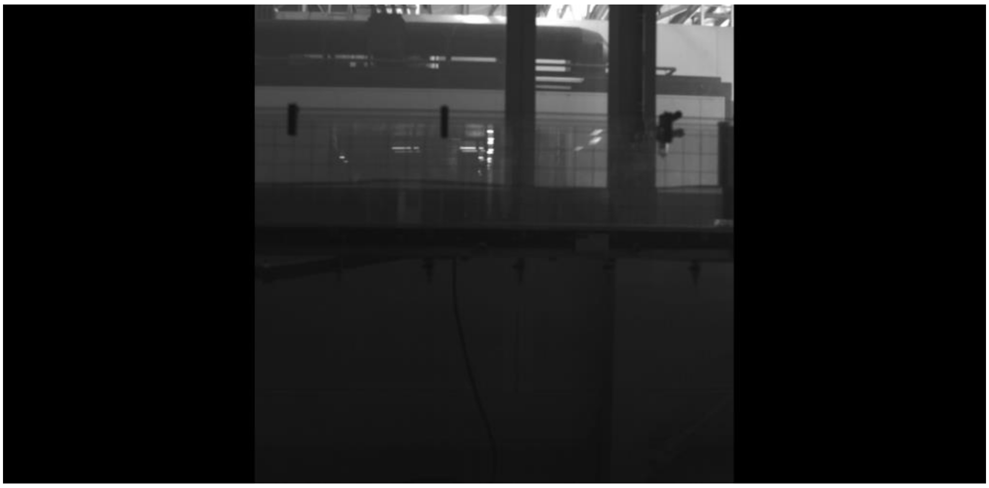
Final Test Screenshots



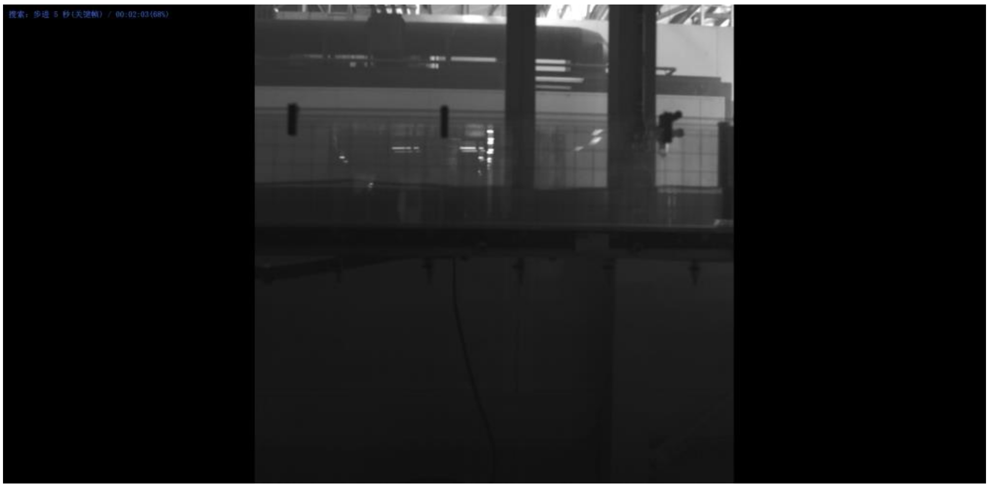
Still water depth = 10 cm, wavelength = 40 cm, wave height = 2 cm



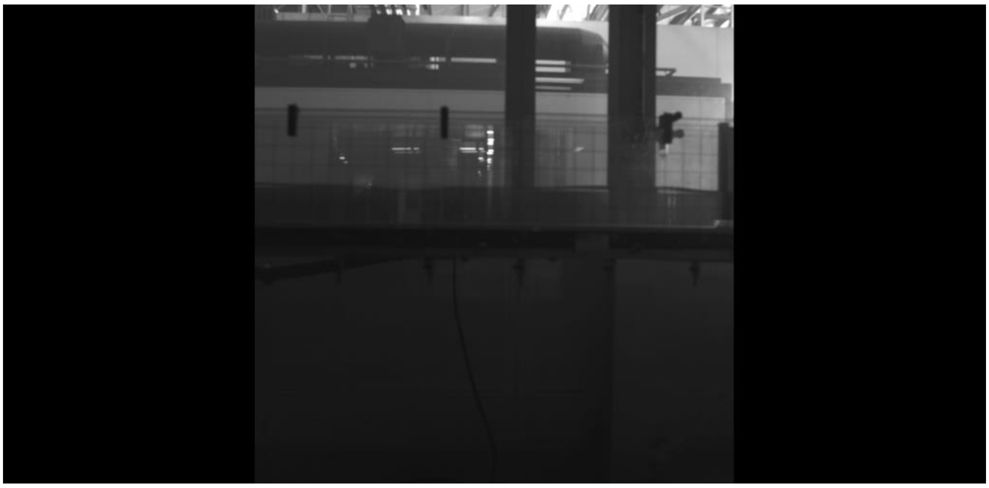
Still water depth = 10 cm, wavelength = 60 cm, wave height = 2 cm



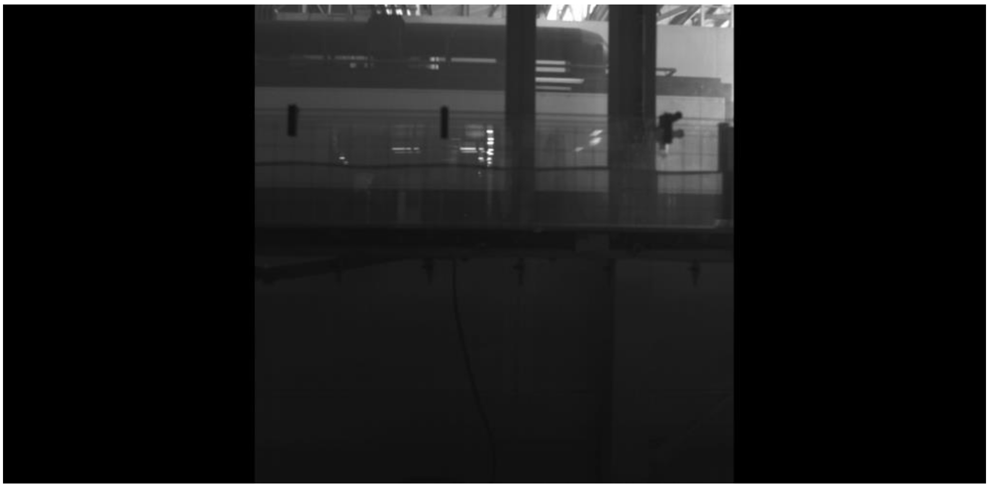
Still water depth = 10 cm, wavelength = 80 cm, wave height = 2 cm



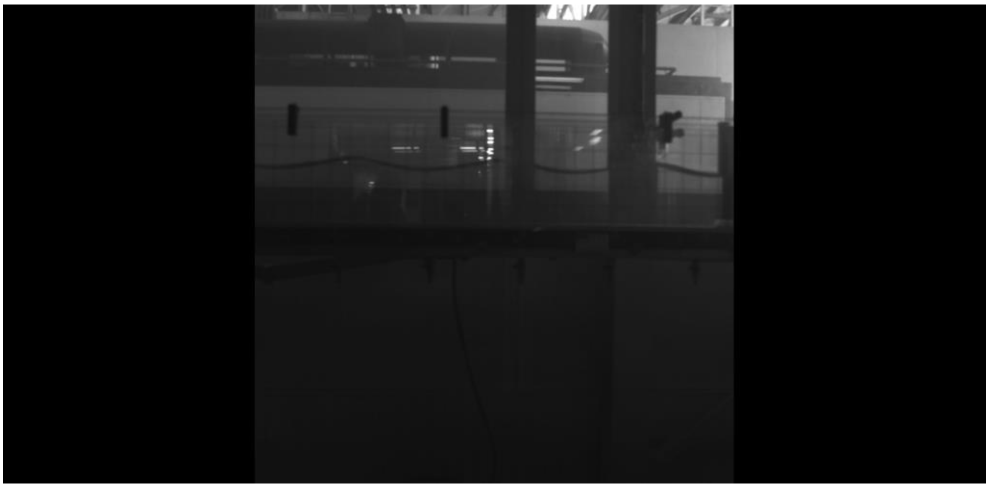
Still water depth = 10 cm, wavelength = 100 cm, wave height = 2 cm



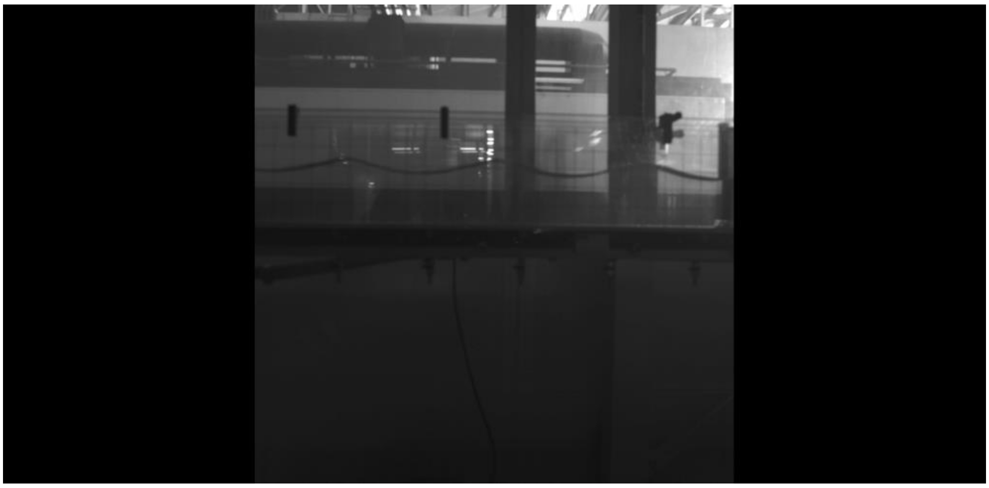
Still water depth = 10 cm, wavelength = 120 cm, wave height = 2 cm



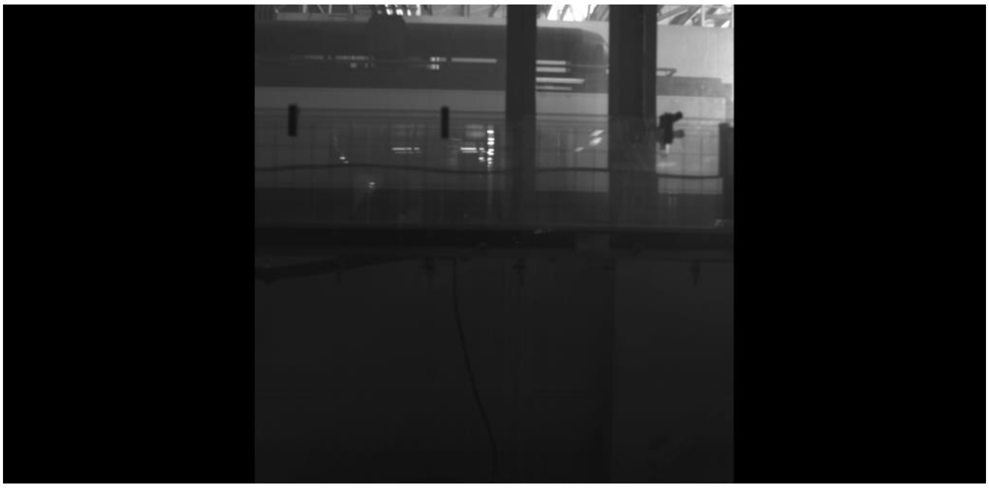
Still water depth = 15 cm, wavelength = 40 cm, wave height = 2 cm



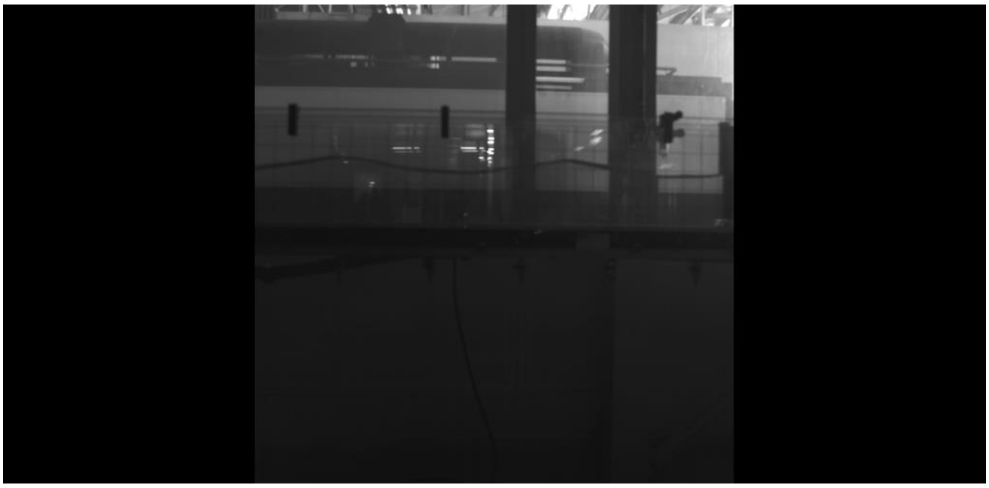
Still water depth = 15 cm, wavelength = 40 cm, wave height = 4 cm



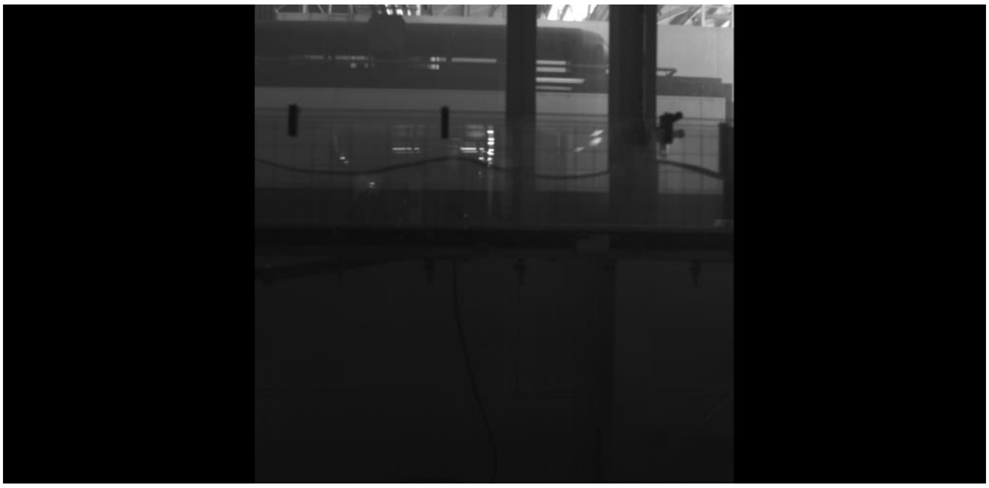
Still water depth = 15 cm, wavelength = 40 cm, wave height = 6 cm



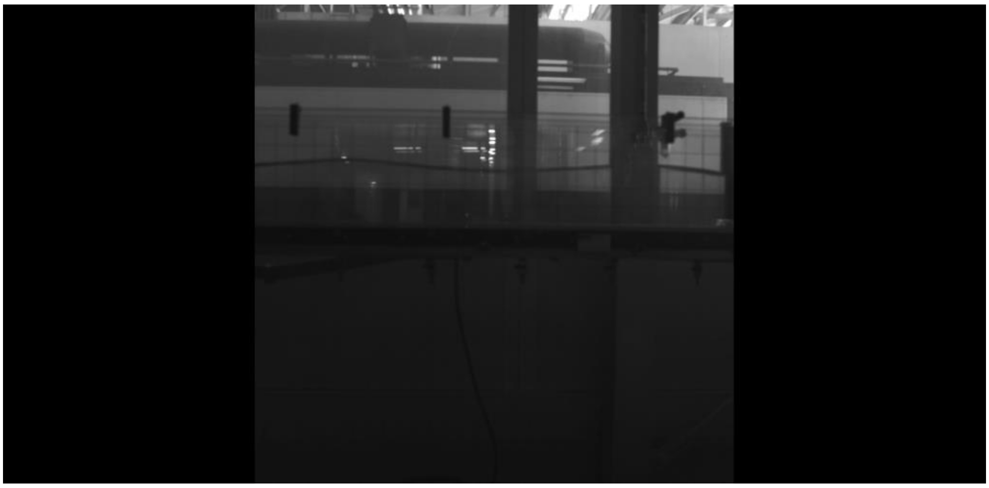
Still water depth = 15 cm, wavelength = 60 cm, wave height = 2 cm



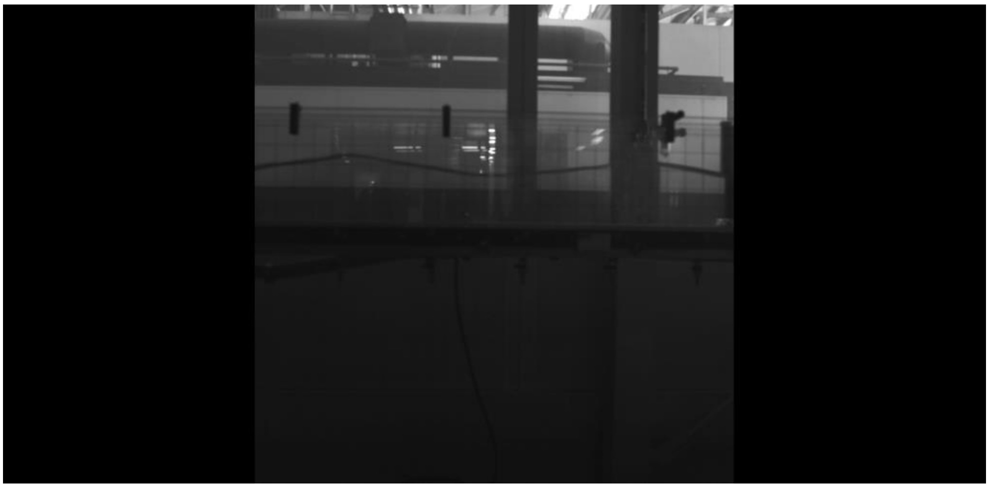
Still water depth = 15 cm, wavelength = 60 cm, wave height = 4 cm



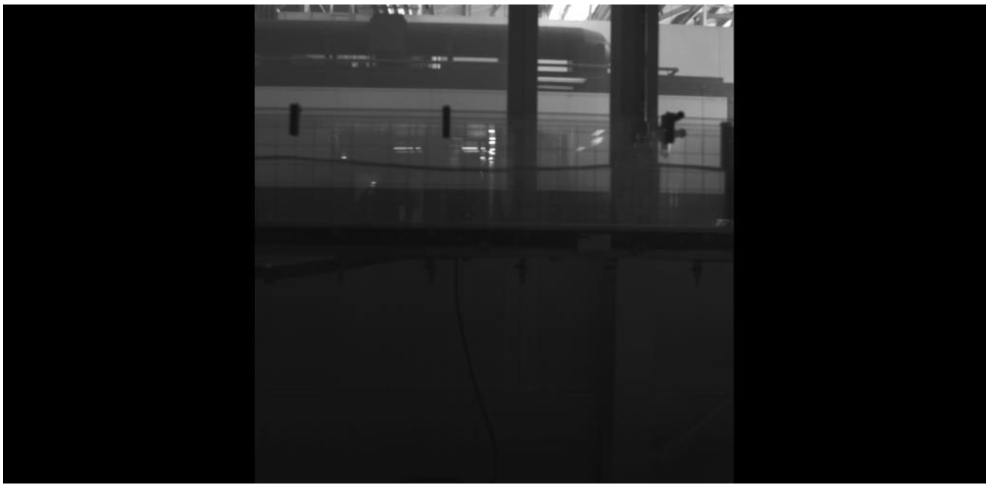
Still water depth = 15 cm, wavelength = 60 cm, wave height = 6 cm



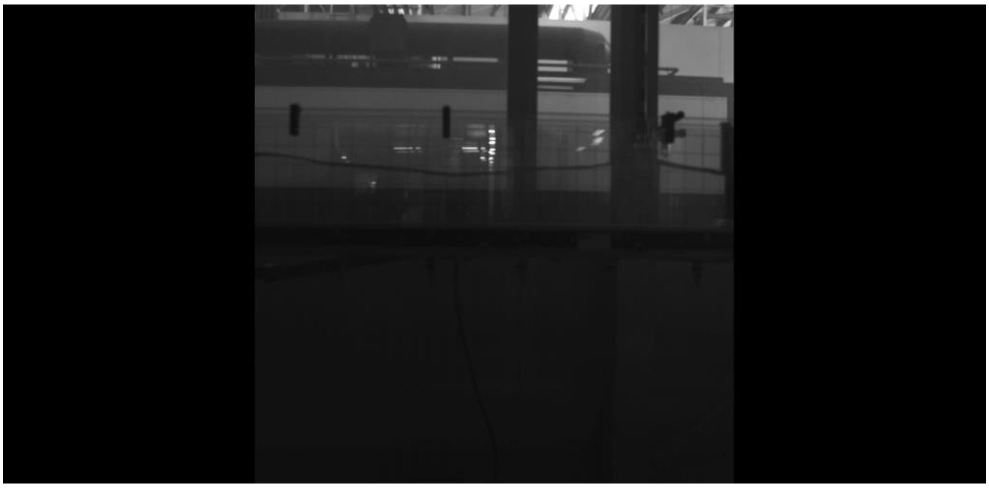
Still water depth = 15 cm, wavelength = 80 cm, wave height = 2 cm



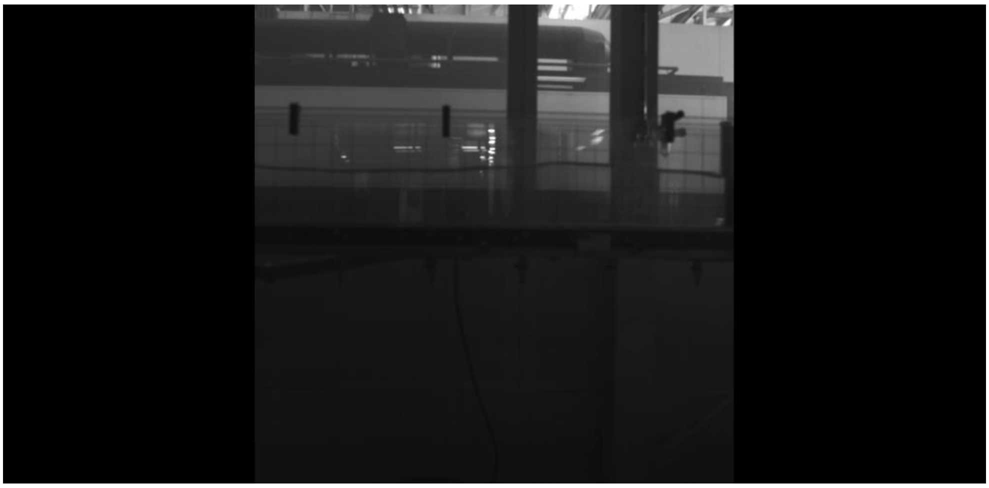
Still water depth = 15 cm, wavelength = 80 cm, wave height = 4 cm



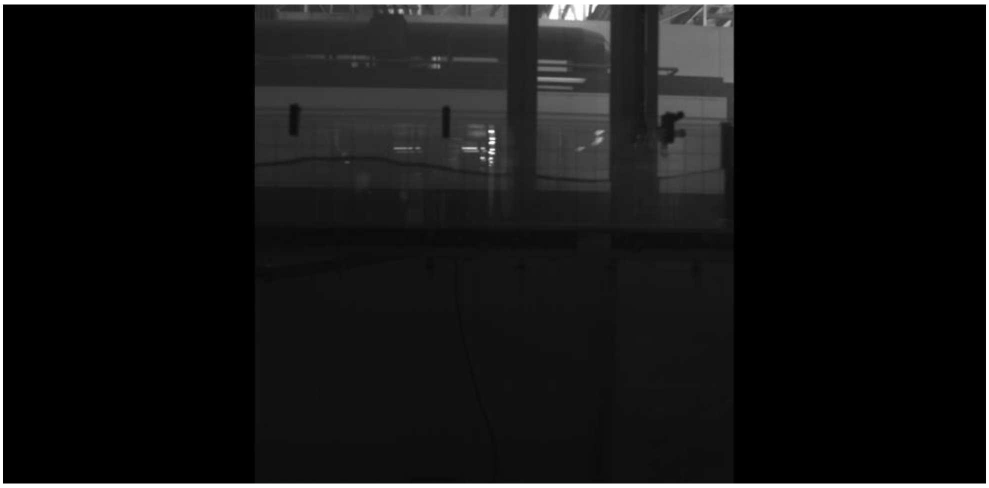
Still water depth = 15 cm, wavelength = 100 cm, wave height = 2 cm



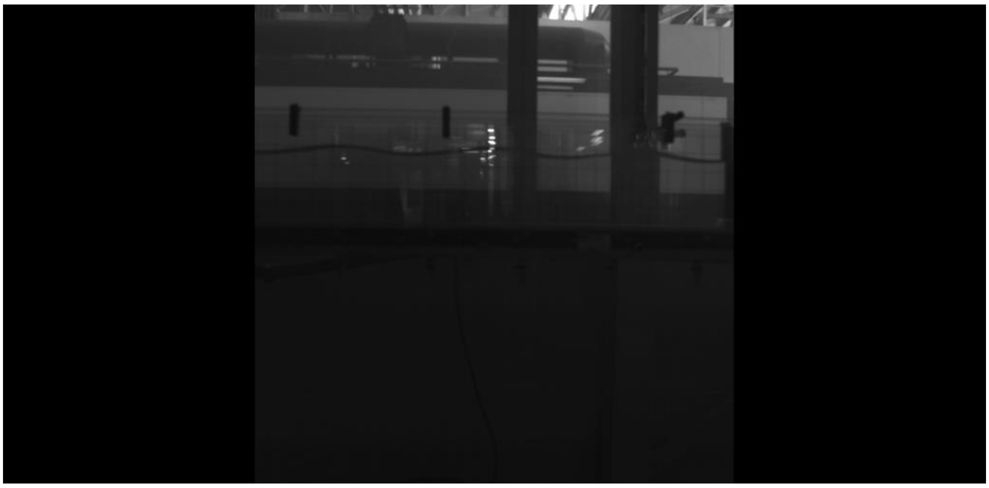
Still water depth = 15 cm, wavelength = 100 cm, wave height = 4 cm



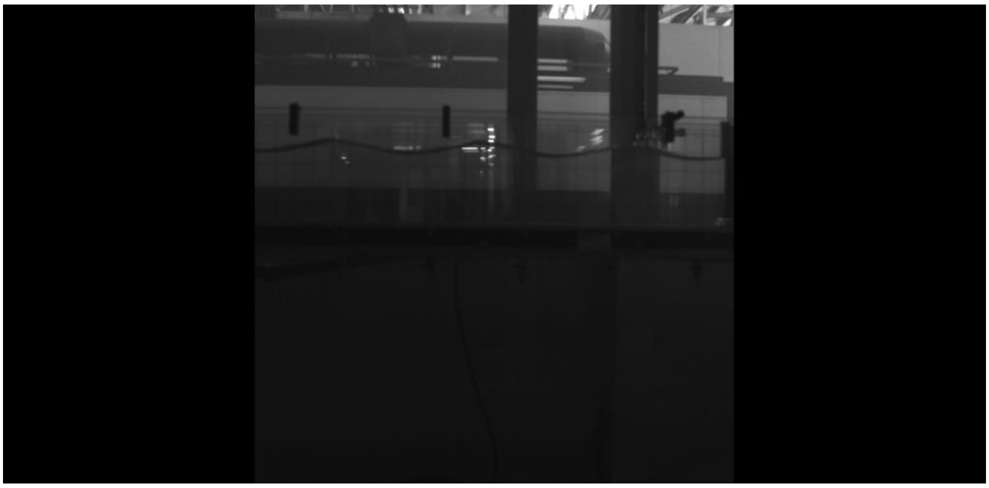
Still water depth = 15 cm, wavelength = 120 cm, wave height = 2 cm



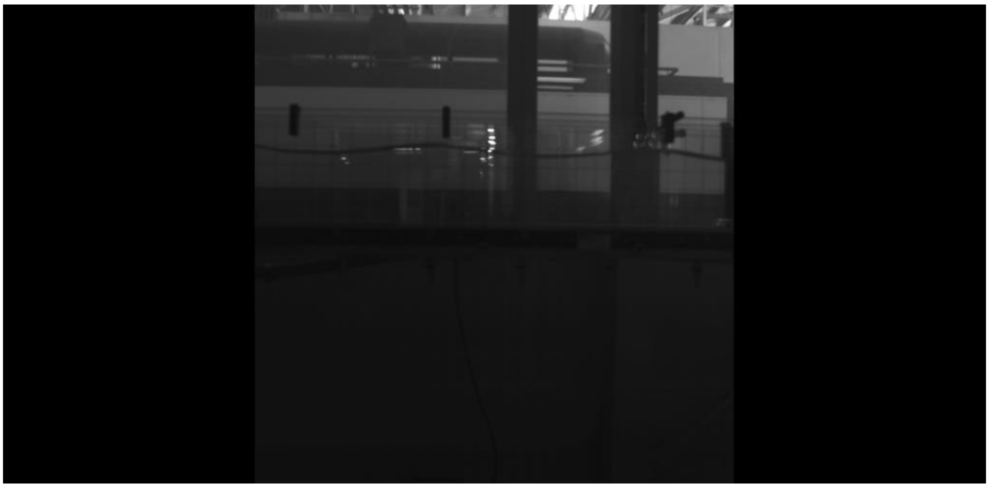
Still water depth = 15 cm, wavelength = 120 cm, wave height = 4 cm



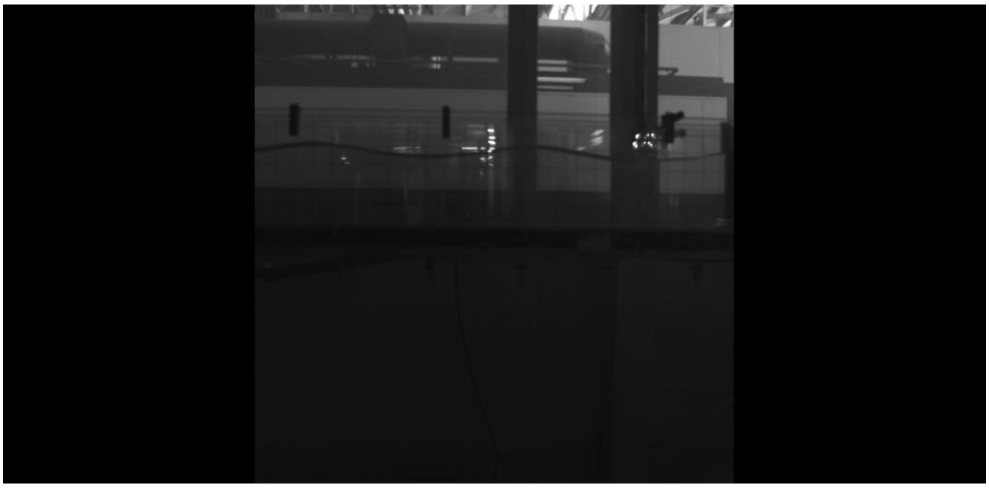
Still water depth = 20 cm, wavelength = 40 cm, wave height = 4 cm



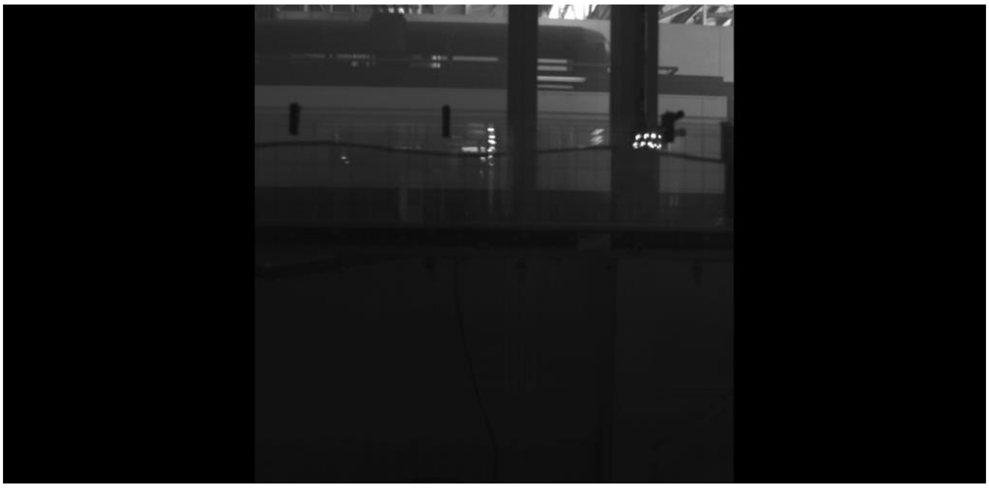
Still water depth = 20 cm, wavelength = 40 cm, wave height = 6 cm



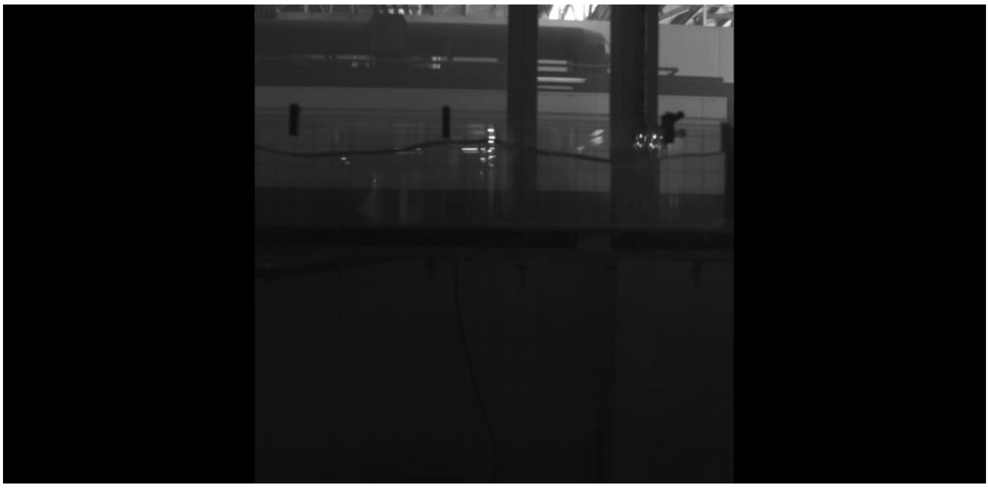
Still water depth = 20 cm, wavelength = 60 cm, wave height = 4 cm



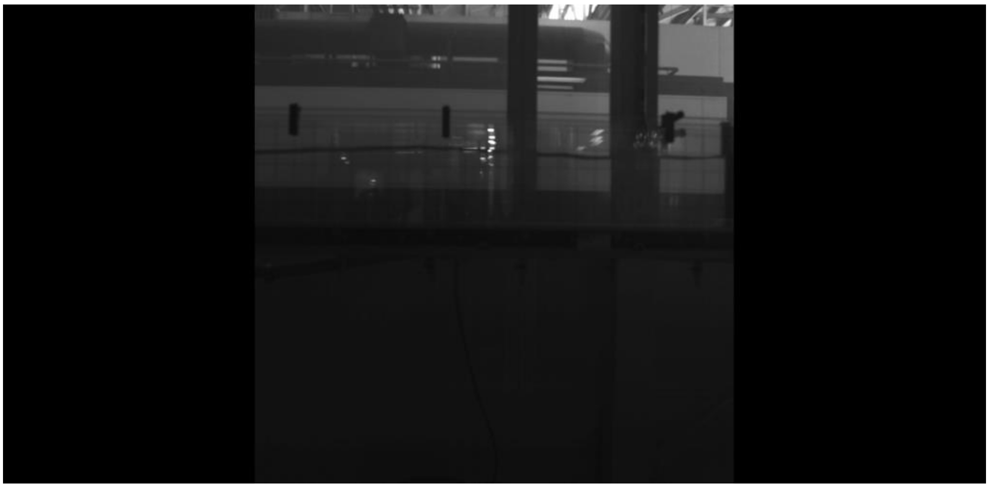
Still water depth = 20 cm, wavelength = 60 cm, wave height = 6 cm



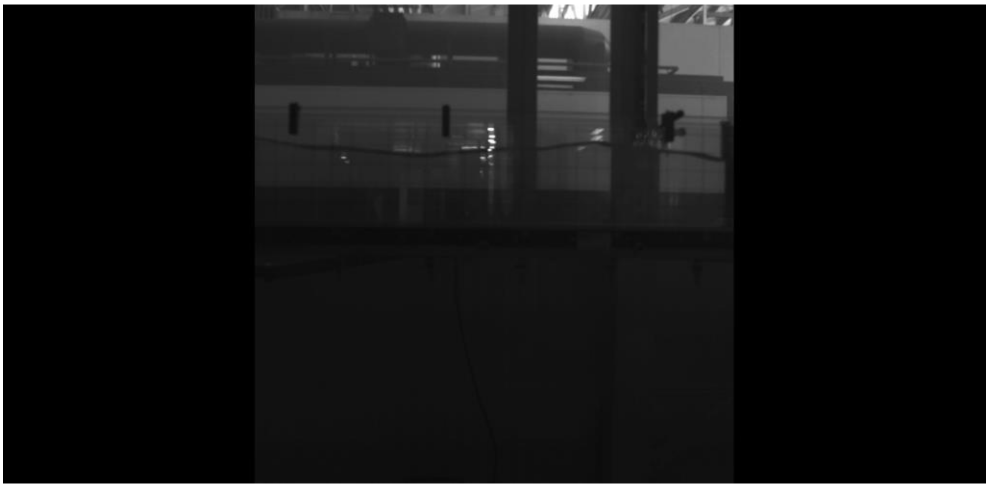
Still water depth = 20 cm, wavelength = 80 cm, wave height = 4 cm



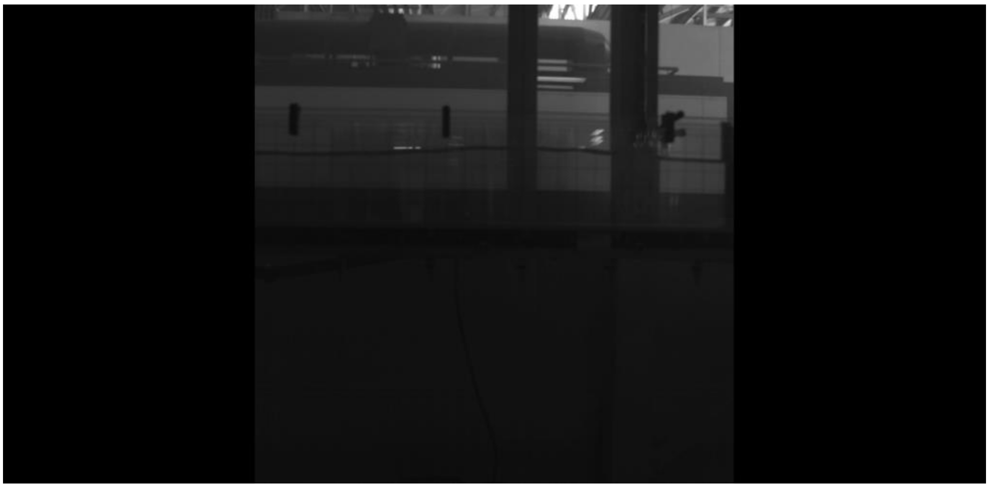
Still water depth = 20 cm, wavelength = 80 cm, wave height = 6 cm



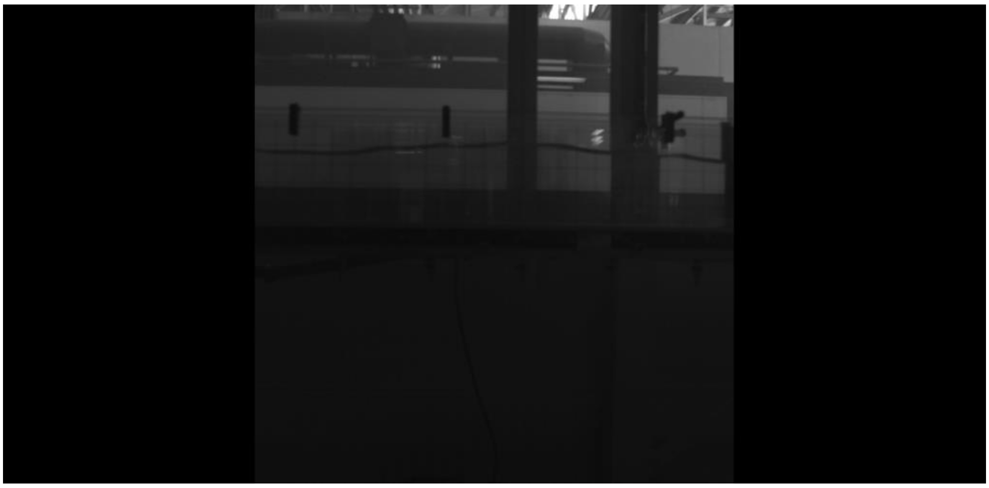
Still water depth = 20 cm, wavelength = 100 cm, wave height = 4 cm



Still water depth = 20 cm, wavelength = 100 cm, wave height = 6 cm



Still water depth = 20 cm, wavelength = 120 cm, wave height = 4 cm



Still water depth = 20 cm, wavelength = 120 cm, wave height = 6 cm