TALLINN UNIVERSITY OF TECHNOLOGY
SCHOOL OF ENGINEERING
Department of Electrical Power Engineering and Mechatronics

SYNCHRONIZATION OF CAMERA AND INERTIAL MEASUREMENT UNIT

KAAMERA JA INERTSIAALSE MÕÕTESEADME SÜNKRONISEERIMINE

MASTER THESIS

Student: Balakrishnan Guruprasath

Student code: 195425MAHM

Supervisors: Jeffrey Tuhtan, Associate Professor
Gert Toming, PhD
Cecilia Monoli, MSc

Co – Supervisor: Even Sekhri, Engineer

Tallinn 2021

(*On the reverse side of title page*)

AUTHOR'S DECLARATION

Hereby I declare, that I have written this thesis independently.
No academic degree has been applied for based on this material. All works, major viewpoints and data of the other authors used in this thesis have been referenced.

"....." ..... 20…..

Author: .....
/signature /

Thesis is in accordance with terms and requirements

"....." ..... 20….

Supervisor: ….....
/signature/

Accepted for defence

"....."..... 20… .

Chairman of theses defence commission: .....
/name and signature/

Non-exclusive licence for reproduction and publication of a graduation thesis[1]

I, Balakrishnan Guruprasath ,

1. grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis Synchronization of Camera and Inertial Measurement Unit

supervised by Jeffrey Tuhtan (Associate Professor), Gert Toming and Cecilia Monoli

1.1 to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;

1.2 to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.

2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.

3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

_____ (date)

[1] *The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.*

Department of Electrical Power Engineering and Mechatronics: Mechatronics and Autonomous Systems Centre

THESIS TASK

Student: Balakrishnan Guruprasath 195425MAHM

Study programme: MAHM02/18 - Mechatronics

main speciality:

Supervisor(s): Jeffrey Tuhtan, Associate Professor

Gert Toming , PhD

Cecilia Monoli, MSc

Co-supervisor: Even Sekhri , Engineer

Thesis topic:

(in English) Synchronization of camera and inertial measurement unit

(in Estonian) Kaamera ja inertsiaalse mõõteseadme sünkroniseerimine

Thesis main objectives:

1. Synchronize image frames with IMU data
2. Find methods to synchronize camera image frames with computer clock
3. Use Python scripts for image capture, image collection and timestamping

Thesis tasks and time schedule:

No	Task description	Deadline
1.	Camera selection and image acquisition	28.02.2021
2.	Delay estimation and development of Python script	31.03.2021
3.	Tests and analysis of results	30.04.2021

Language: English **Deadline for submission of thesis:** 18 May 2021

Student: Balakrishnan Guruprasath "....."……..20….a

/signature/

Supervisor: Jeffrey Tuhtan "....."…….20….a

Gert Toming

Cecilia Monoli

/signature/

Head of study programme: Mart Tamre "....."…….20…..a

/signature/

Terms of thesis closed defence and/or restricted access conditions to be formulated on the reverse side

TABLE OF CONTENTS

List of abbreviations and symbols 6
List of Figures 7
List of Tables 9
1. INTRODUCTION 10
2. LITERATURE OVERVIEW 13
2.1 Objectives of This Thesis 16
3. HARDWARE AND SOFTWARE 18
3.1 Camera 18
3.2 Wearable Sensing Unit 21
3.3 EMP Coil 22
3.4 Arduino UNO 24
3.5 Python 25
4. METHOD 26
4.1 Image acquisition 26
4.2 Data Acquisition 28
4.3 Python Script 29
4.4 Arduino code 31
5. TESTS AND RESULTS 32
5.1 Image capture delays 32
5.2 Drop Test 36
5.3 Gait Cycle Test 42
SUMMARY 52
KOKKUVÕTE 54
LIST OF REFERENCES 56

List of abbreviations and symbols

No	Abbreviations	Explanation
1	IMU	Inertial Measurement Unit
2	SD card	Secure Digital Card
3	RGB	Red, Green and Blue
4	Ref. No	Reference Number
5	PC	Personal Computer
6	IoT	Internet of Things
7	LED	Light Emitting Diode
8	EMP	Electro-magnetic Pulse

List of Figures

Figure 2. 1 Evaluation of transmission time [11] 14
Figure 3. 1 uEye Camera used in this thesis 18
Figure 3. 2 Canyon C3 web camera 19
Figure 3. 3 Wearable sensor unit 21
Figure 3. 4 Electromagnetic pulse coil 23
Figure 3. 5 General circuit plan of EMP coil 23
Figure 3. 6 EMP coil signal 24
Figure 3. 7 Arduino Uno connected to EMP coil 24
Figure 4. 1 uEye Cockpit user interface 26
Figure 4. 2 Wearable sensor unit data displayed in an Excel sheet 28
Figure 4. 3 Spike in Magnetic Strength reading (Mag Z) 29
Figure 4. 4 Image capture and EMP coil activation script flowchart 30
Figure 4. 5 Code written into the Arduino Board using Arduino IDE 31
Figure 5. 1 Delay approximation setup 32
Figure 5. 2 System time displayed in real time 33
Figure 5. 3 Image 'ex322' 34
Figure 5. 4 Image timestamps along with the difference 35
Figure 5. 5 Sensor unit attached to a box for test 36
Figure 5. 6 Time difference between the 1st and 2nd images 37
Figure 5. 7 Image 'ex0' and the left and image 'ex1' on the right 37
Figure 5. 8 Magnetic field strength plot 38
Figure 5. 9 EMP signal register point (Data no. 1672) 38
Figure 5. 10 Timestamps added to the sensor unit data 39
Figure 5. 11 Data no vs acceleration plot 39
Figure 5. 12 The point at which the acceleration values of Z axis start reducing 40
Figure 5. 13 The point at which the acceleration values of z axis reaches back to the nominal value 40
Figure 5. 14 Image 'ex363' on the left and image 'ex381' on the right 41
Figure 5. 15 Image timestamps 41
Figure 5. 16 Sensor units attached to the thigh and lower leg 42
Figure 5. 17 EMP synchronization coil setup 43
Figure 5. 18 Magnetic field strength plot of the top sensor 44
Figure 5. 19 Acceleration plot of the top sensor 44
Figure 5. 20 'ex613' on the left and 'ex705' on the right 45
Figure 5. 21 Some intermediate image frames 45
Figure 5. 22 Timestamps of image frames 'ex613' and 'ex705' 46
Figure 5. 23 Magnetic field strength spike of bottom sensor 47

Figure 5. 24 Acceleration plot of the bottom sensor 47
Figure 5. 25 Acceleration data of both sensor units after synchronization.....48

List of Tables

Table 2. 1 Previous work 15
Table 3. 1 Camera Specifications 18
Table 3. 2 PENTAX C1614-M Specifications 19
Table 3. 3 Canyon C3 Specifications 20
Table 3. 4 Differences in cameras 20
Table 3. 5 Python libraries used 25
Table 5. 1 Timestamps of sensor data and image frames compared 41
Table 5. 2 Top sensor and image frame timestamps compared 46
Table 5. 3 Bottom sensor and image frame timestamps compared 48
Table 5. 4 Comparison of timestamps from further tests 49

1. INTRODUCTION

Humans living in the 21st century, are highly reliant on technology to carry out almost any task. From the time we wake up in the morning to the time we get into bed, we use modern devices to make our lives easier. These devices employ sensors which measure physical environment and convert the measurements to data, interpreted by humans or by the devices themselves. Most of these devices employ more than one sensor so that multiple parameters can be measured and thereby provide a more complete picture of the environment to the interpreter.

Data from all the relevant sensors must be aligned with respect to some common quantity so that the interpreter can make clear and accurate decisions based on them. This "common quantity" is usually time. Synchronizing various data sets with respect to time is straightforward in cases where all the sensors share the same controller and the same clock. This is because the controller will be able to trigger the sensors to start and stop collecting the data at the same time. Since the sensors also share the same clock, additional processing of data for data synchronization or data fusion will be redundant.

Differently, synchronization of data from sensors controlled by independent controllers with their own local clocks is not straightforward. This is due to issues like trigger delays and local clock drift [1]. Trigger delay is the time delay between a trigger and the moment data collection starts. Clock drift is the gradual variation of a local clock compared to a reference clock. Also, since most current devices are also wireless, transmission time, as time required to send data from the sensor node to the final node or processing computer becomes a factor too. Therefore, while processing sensor data for data fusion, the above-mentioned issues have to be accounted for.

Literature on such problems barely available and this thesis aims to address one such technology gap. The main aim of this thesis is to synchronize video from camera to wireless sensor (IMU – Inertial Measurement Unit) data which is not transmitted in real time. This is known as offline data synchronization. In instances where wireless sensors are used underwater, normal transmission of data via radio waves is not possible. This is due to the interaction of electromagnetic waves and water [2] and as a result leads to path loss, variation in velocity of propagation and absorption loss as detailed in [3].The is also the reason why most underwater communications use acoustic waves to communicate [4][5], especially in saltwater. Therefore, the sensor data will have to be collected in storage devices such as SD cards (secure digital card) and later retrieved to be analysed.

The result of this thesis will be especially useful in the medical industry. Specifically, for gait analysis [6], the science of studying the biomechanics of motion during walking. Currently, it is carried out using different type of equipment aiming at the definition and evaluation of motion impairments, pathologies and injuries development. Underwater Gait analysis is especially useful as rehabilitation exercise, for patients who have problems in movement of their lower limbs either due to illness or old age [7] or due to accidents. This is because an underwater environment can mimic lower gravity and therefore can help patients begin their recovery without having to carry their entire body weight.

Usually, the results of gait analysis (limb angle, position and acceleration) are plotted into a graph and is analysed by the medical professionals or a video feed is obtained (via regular cameras or specialized ones) to study the motion. Since the video feed is in 2D, the motion might be oversimplified as depth perception is not available. But by synchronizing both the video feed / image frames along with the sensor data, without expensive or complicated systems, can make the analysis of obtained movement data less complicated and less time consuming. This can also fill in the limits of each other (IMU data and video feed) to provide a more accurate picture of limb movement. This can also be used to help patients understand movement of their limbs without much difficulty.

In this thesis the apparatus (camera and IMU sensors) used are from different manufacturers and therefore the clock in the camera and the internal clocks in the sensors are different. As the trigger mechanism for the sensors and the camera are different, there is no simple way to synchronize the image frames with the sensor data. Here, the station time (computer time) is used as the reference time (reference clock) and synchronize the image frames from camera clock to the reference clock. Next, once the data from the sensor have been obtained, synchronize the data to the reference clock and thereby synchronize it to the obtained image frames. Synchronization of image frames to sensor data will also require the approximation of camera exposure time[8], transmission time and possible delays within the camera itself.

The methodology developed, tested and validated in this thesis is based on a seven-stage workflow as follows:

1. A camera, which can obtain images with trigger signals from a Python script was selected.

2. Experiment was conducted to calculate the approximate delay time (exposure time and transmission time) of the images from the camera to computer.
3. The transmitted images were named sequentially and timestamped with the local computer time.
4. The recorded sensor data were transferred to the computer and image timestamps were matched to sensor data time.
5. Finally, the images were synchronized to corresponding sensor data.
6. Tests were conducted to ensure the image- data synchronization is within accepted limits.

2. LITERATURE OVERVIEW

Most of the recent work exploited wireless sensors which transmit data to the local computer in real time, as it is showed in the literature review resumed in Table 2.1 . The algorithm proposed in [9] is to synchronise the data from the sensors rather than synchronize all sensor node clocks. The authors have considered this method because of various factors that can affect sensor node clock synchronization such as clock drift. Specifically, IoT devices can be used in remote and extreme environments, mechanical vibrations, temperature fluctuations, humidity and other factors can de-synchronize node clocks. Also due to the use of IoT devices in remote environments, the authors have assumed low range of network and therefore the data has to be passed from one sensor node to the other until it reaches the sink node (computer). The algorithm developed here, therefore, is highly reliant on residual time (the particular time period a data packet remains within a node), no of steps (nodes) between the sensor node and the sink node and average skew deviations, and therefore is not suitable for the cases discussed in this thesis.

Bluetooth network for data transfer and mobile phone for data processing has been used in [10]. The main goal of this paper was to minimize errors in synchronizing data sent from various sensors to a mobile phone via Bluetooth. Here too, like the previous paper, synchronization of data was preferred to synchronization of sensor clocks. The proposed algorithm in this paper depends on Bluetooth transfer delays and timestamping delay. This paper also deals with real time data fusion unlike this thesis which is offline (not real time data synchronization) and no Bluetooth network is used.

[11] is concerned the synchronization of RGB depth camera and wearable IMU sensor data that can be integrated in ambient assisted living applications. Essentially, it allows healthcare professionals to monitor disabled or old patients in their home environment. This, is the closest work that has been done related to the aim of this thesis. The RGB depth camera used by the authors of was the Microsoft Kinect and the inertial measurement unit here was manufactured by Shimmer Research. The data from the IMU were sent to the local computer via Bluetooth. The local clock of the RGBD camera was not accessible and therefore the authors timestamped the image frames when the images were received by the computer. This requires estimation of exposure (the amount of time the sensing device is open) and transfer delays (the time needed to encapsulate and transmit the image frame from the camera to the computer) and the authors have estimated them via an Arduino board controlling LEDs. The authors have used a variant of Cristian's algorithm[12] to synchronize the Shimmer IMU sensor and

the computer. The same algorithm couldn't be used to synchronize the RGB depth camera and the computer because the camera's local clock was not accessible.

The Arduino board and the LEDs used in the above paper worked in the following way to estimate transmission time. The Arduino board was connected to the same PC that was connected to the RGB depth camera and controls 7 LEDs. When the PC received the 1st frame (F0), a timestamp (t_{0_PC}) was set and a command was sent to the Arduino board. The Arduino board then waited for 20 milliseconds and turned on the LEDs sequentially with a delay of 3 milliseconds. When frame F2 was received by the PC, a timestamp (t_{2_PC}) was set and by counting the number of LEDs on in the frame, the time t_{2_K} was calculated. The transmission time is then obtained by finding the difference between t_{2_pc} and t_{2_K}. This is illustrated in the image below. The 20ms delay used in the Arduino board was to centre the on/off LEDS at time t_{2_K}. The same test was carried out around 75 times and the average transmission time was chosen.

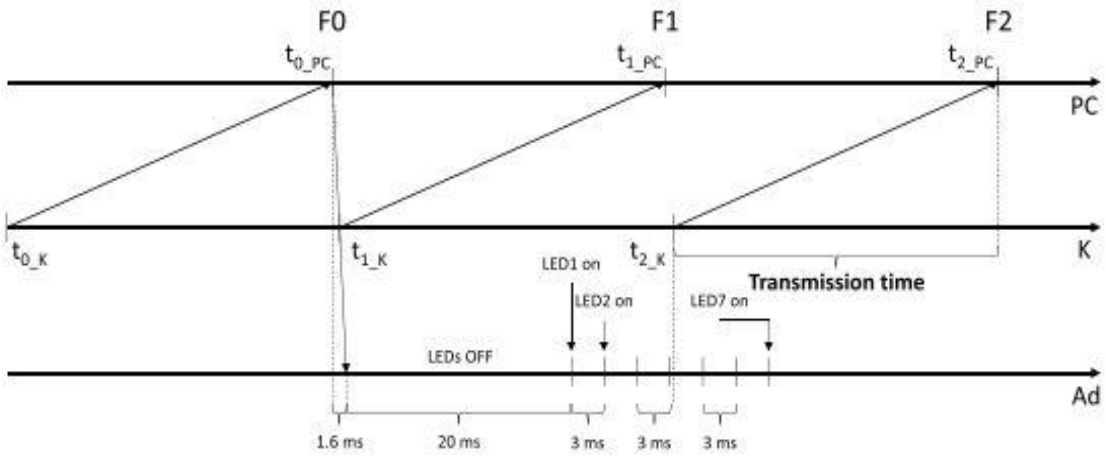

Figure 2. 1 Evaluation of transmission time [11]

Very little work has been done similar to [11] which involves synchronization of Camera frames and IMU data. Other papers [13][14][15] discussing Camera-IMU systems usually are interested in spatial and temporal calibration of the sensors. They typically employ predictive algorithms such as Kalman Filters and Extended Kalman filters[16] to calibrate their position in space. The authors of [13] used employ a visual-inertial sensor unit where the triggering of both the camera and the IMU was done by the same controller. Therefore, even though the camera and IMU have separate clocks, synchronization was done in a relatively straightforward manner. The data coming in from the sensor and the camera were timestamped according to delays based on the methods proposed in [17].

Lastly, research dealing with data synchronization of wireless sensors in an underwater environment is virtually non-existent. Most of the research done in underwater environments deal with communication and advanced data acquisition methods[4][5] rather than data fusion or data synchronization.

Table 2. 1 Previous work

Ref.No	Date	Paper	Aim	Approach/ Solution
[9]	2019	Synchronization of data measurements in wireless sensor networks for IoT applications	Algorithm to synchronize data from wireless sensors rather than synchronizing node clocks	Algorithm for wireless sensors transmitting data synchronization Deals with multiple nodes and takes into account the transmission time/distance to the sink node
[10]	2013	A novel approach to multi-sensor data synchronization using mobile phones	Algorithm to synchronize data received from multiple sensors via Bluetooth to mobile phone	Data transmission through wireless network (Bluetooth). Involves delays associated to Bluetooth network
[11]	2015	Time Synchronization and Data Fusion for RGB-Depth Cameras and Inertial Sensors in AAL Applications	Synchronization of the data captured from RGB-Depth cameras and wearable inertial sensors	The RGBD camera is used to estimate patient positions IMU is synchronized to PC using Cristian's Algorithm. Camera exposure and transmission time found using Arduino with LEDs
[13]	2014	A Synchronized Visual-Inertial Sensor System with FPGA Pre-Processing for Accurate Real-Time SLAM	Development of a visual inertial sensor unit to be deployed in robots for simultaneous localization and mapping capabilities	Simultaneous triggering of camera and IMU Single processor/controller
[14]	2010	Visual-Inertial Sensor Fusion: Localization, Mapping and Sensor-to-Sensor Self-calibration	To obtain an algorithm, based on unscented Kalman Filter, for self-calibration of the transform between a camera and an IMU.	Algorithm based on Kalman Filter mainly for robot navigation. Sensor fusion rather than data fusion

Table 2. 2 Previous work …. continued

Ref.No	Date	Paper	Aim	Approach/ Solution
[15]	2014	Online temporal calibration for camera-IMU systems: Theory and algorithms	Algorithm for temporal calibration of camera and IMU.	Vision aided inertial navigation. Focuses on offset time estimation and mainly 3d pose estimation from the camera- IMU data. Uses the offset time along with EKF state vectors to calculate offset.
[17]	2010	Unified Temporal and Spatial Calibration for Multi-Sensor Systems	Estimate the temporal offset between measurements of different sensors and their spatial displacements with respect to each other	Offset time method to synchronize multiple sensors in space

2.1 Objectives of This Thesis

As mentioned above in the Literature Overview section, the amount of research/ papers on offline synchronization of camera images and wireless IMU sensor data is limited. Most of the papers dealt with wireless sensors and real time data synchronization. The methods proposed in those papers are not suitable for this thesis as they do not deal with both real time data and offline data at the same time. Furthermore, no previous works were found on the specific topic of underwater sensor data and camera synchronization.

Cippitelli's study [11], overlaps fairly the objective and methodology of the current Thesis work. Specifically, the RGB depth camera images were sent to the PC in a similar way and a similar technique to estimate the average transmission time of images can be used. Differently, since the camera used in this thesis employs user controllable exposure settings, the method used to estimate exposure time in [11] is not required. Concerning the sensors, the mentioned study exploits Shimmer IMU sensor with Bluetooth network to transmit data in real time to the PC using a variant of Cristian's algorithm. Dissimilarly, the IMU sensors in this thesis are developed for underwater purposes and cannot transmit data in real time, a different approach must be taken to synchronize the IMU sensor data to the PC clock.

None of the above-mentioned papers use Python script to synchronize sensor data with image frames. The decision to use Python code was made to guarantee flexibility and versatility of the developed system, so that any camera with a Python library or capability of interacting with a Python script can be used. This makes replacement of the camera used in this thesis with any other camera having a Python library easy.

The above-mentioned technological and research gaps, which are also the objectives of this thesis, are summarized below:

- **Objective 1:** Synchronize images from a camera (transmitted in real time to the computer) with data from IMU sensors which are not transmitted to the computer in real time.

- **Objective 2:** Experiment with various methods to synchronize the image frames with the computer clock / reference clock.

- **Objective 3:** Use a Python script to do camera setup, data collection, timestamping and synchronization.

- **Objective 4:** Make the Python script versatile enough for change in cameras without major changes needed to the script.

3. HARDWARE AND SOFTWARE

3.1 Camera

The camera initially used in this thesis is a uEye UI-5240HE-M-GL, as shown in Figure 3.1 .

Figure 3. 1 uEye Camera used in this thesis

It is an industrial highspeed monochrome camera with a CMOS sensor. The specifications of the camera are provided in the table 3.1

Table 3. 1 Camera Specifications

Name	**Specification**
Model	UI-5240HE-M-GL
Sensor	0.5'' CMOS
Resolution	1.3 Megapixel (1280x1024)
Maximum frame rate	60 frames per second
Colour	Monochrome
Video Output	Gigabit Ethernet

The images obtained by the camera are transmitted to the computer via an Ethernet cable. The lens used with this camera is PENTAX C1614-M. This lens is controlled manually and is mainly used for image processing applications. Some benefits of using this lens are manual adjustability, high light intensity and lower distortion. Some specifications of this lens are given in the table below.

Table 3. 2 PENTAX C1614-M Specifications

Name	**Specification**
Format	2/3”
Focal Length	16.0 mm
Aperture Range	F1.4- 16
Iris Control	Manual
Focus Control	Manual
Minimum Object Distance	0.25 m

This uEye camera was chosen for its high frame rate which is closer to the data acquisition rate of the sensing unit. But due to complications with the Python library, explained further in the method section (4.1 Image Acquisition), this camera had to be replaced with a different camera. The replacement camera chosen was Canyon C3 web camera.

Figure 3. 2 Canyon C3 web camera

Table 3. 3 Canyon C3 Specifications

Name	Specification
Model	CNE-CWC3N
Sensor	¼'' CMOS
Resolution	Max. 12 Mega Pixel (1280x720)
Maximum frame rate	30 frames per second
Interface	USB 2.0

There were several key differences between this web camera over the highspeed uEye camera. They are listed in the table below.

Table 3. 4 Differences in cameras

uEye Camera	Canyon C3 webcam
Ethernet connection to computer	USB connection to computer
High speed camera	Standard webcam
1280x1024 Maximum resolution	1280x720 Maximum resolution
Heavy	Lightweight
Python library available	Python library unavailable
Fixed	360-degree rotatable view

As shown in the table above, there were some advantages and disadvantages in using the web camera over the uEye high speed camera. The image quality was higher in the uEye camera compared to the web camera but this is not an indispensable issue as the image quality of the web camera was clear enough to continue with tests. Also, since the image size of the images obtained from the web camera were smaller, the change from Ethernet connection, which has a higher throughput, to USB connection didn't affect any part of this thesis. The only advantages the uEye camera had over the webcam were the high framerate capability and the Python library. The Python library

would enable much more control of the camera but due to reasons explained in the methods section of this thesis, this advantage was not very helpful.

The webcam has some minor advantages over the uEye camera. Since it is lightweight and rotatable, performing tests in various environments and angles become easier and not cumbersome. Also, the webcam can be connected to any computer with a USB port and easily accessed using Python libraries such as OpenCV to make use of basic functionalities. The uEye camera required a lot of setup code within the Python script and also required network port settings to be changed. The 'pyueye' Python library does not come with any documentation and therefore would be difficult for an inexperienced user to use many of the available functions.

3.2 Wearable Sensing Unit

The wearable sensing unit, named TinyTag, used in this thesis was designed by the Center for Biorobotics of Tallinn University of Technology to be used in rehabilitation clinics for underwater human and animal gait analysis. This unit consists of several parts including the IMU. The unit itself, excluding the battery, is approximately 30mm in length and 12mm in width. This unit without the battery is shown in the figure 3.3.

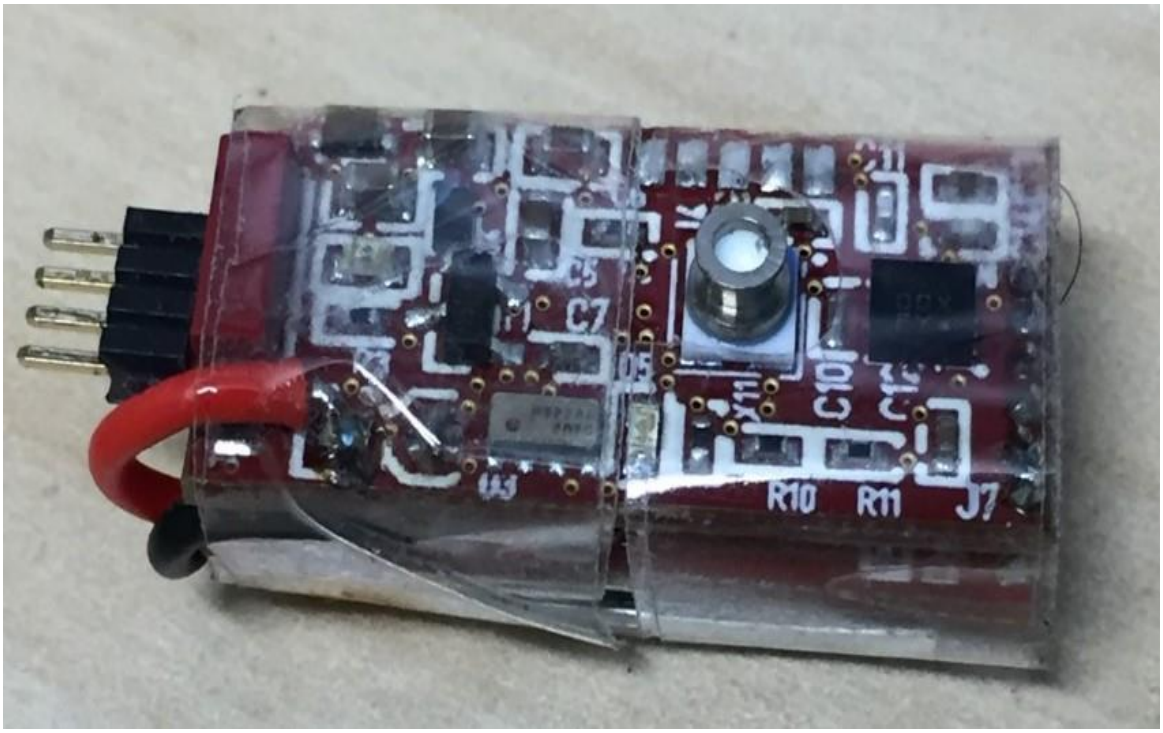

Figure 3. 3 Wearable sensor unit

The key components of this unit are listed below.

1. 32 bit ARM® Cortex®-M0+ SAM D21G microcontroller (Microchip) [18]
2. Absolute orientation sensor BMX160 (Bosch) [19]
3. Absolute pressure sensor MS5837-02BA [20]
4. Real time-clock RV-3028-C7 [21]
5. MicroSD card [22]
6. 50mAh lithium polymer battery [23]

The IMU unit used for this thesis (BOSCH BMX 160) is a low power nine axis sensor consisting of a triaxial accelerometer, a triaxial gyroscope and a geomagnetic sensor. The accelerometer is used measure the rate of change of velocity. This acceleration is always measured relative to gravitational acceleration. Therefore, an acceleration of zero means that the sensor is moving at a constant velocity or is at rest. The next component in an IMU is the tri axial gyroscope that measures the angular velocity. This is also used to maintain orientation of an object. The BMX 160 IMU outputs the angular velocity in the units of degrees per second. Finally, the geomagnetic sensor measures and outputs the magnetic field density and can be used to locate the north and south poles and therefore the orientation.

The TinyTag device is turned on by coming into contact with an external magnet. It can also turn itself on automatically by pre-programming the start date and time. The device can be turned off using the magnet or by pre-setting a operational time. Ths sensor unit is equipped with a standard 1.27mm pitch header row with 4 pins and can be connected to a computer to transfer data, set time and date, set turn on and off time and also charge the 50mAh battery.The device is waterproof and is electrically isolated so that it doesn’t get into contact with human or animal body while performing tests. It is also coated with a mixture of epoxy resin and polamide partices Vestosint 1101.

3.3 EMP Coil

Another instrument used in this thesis was an EMP (electromagnetic pulse) coil designed by the Center for Biorobotics of Tallinn University of Technology. This coil, shown in figure 3.4 , is used to create an electromagnetic pulse that can be clearly measured by the magnetic sensors in the IMU. The spikes in magnetic readings can later be used to synchronize multiple sensors with each other.

Figure 3. 4 Electromagnetic pulse coil

This pulse helps to synchronize the multiple wearable sensing unit measurements used in this thesis with each other. The coil circuit diagram and the pulse generated when the Test button or and external signal is given, is shown in the figures 3.5 and 3.6.

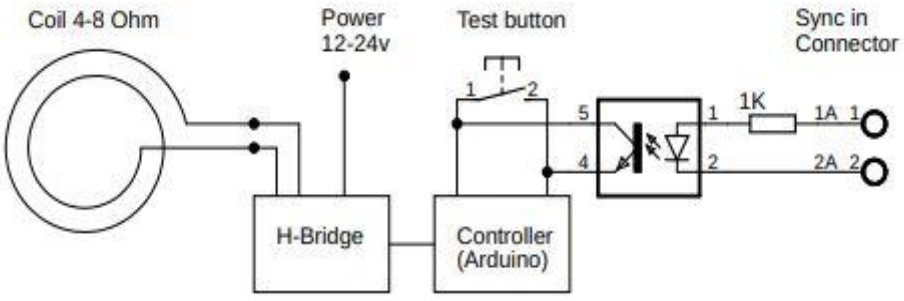

Figure 3. 5 General circuit plan of EMP coil

The trigger signal for the pulse is provided by an Arduino UNO board which is controlled by a Python script. The 5V output from the Arduino board is connected to the "Sync in connector" (shown in the figure 3.5) part of the coil.

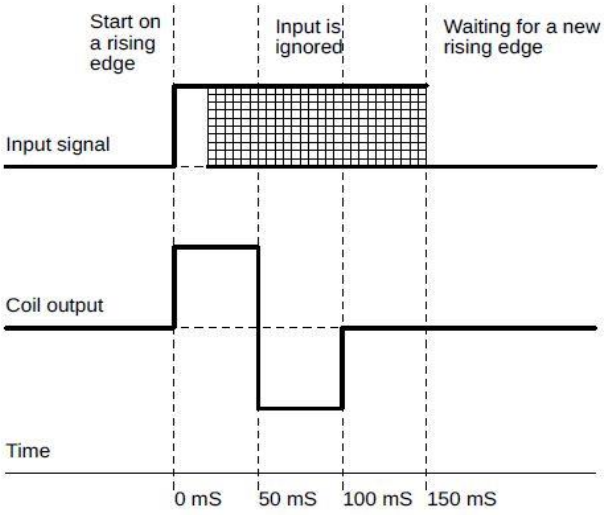

Figure 3. 6 EMP coil signal

As shown in the above figure 3.6, the pulse lasts for 100ms and switches polarity at 50ms. The intensity of the signal registered depends on how close the sensor is to the EMP coil and also the orientation. The Arduino signal to the coil doesn't have to be turned off because the EMP signal is only triggered at the rising edge of the input signal.

3.4 Arduino UNO

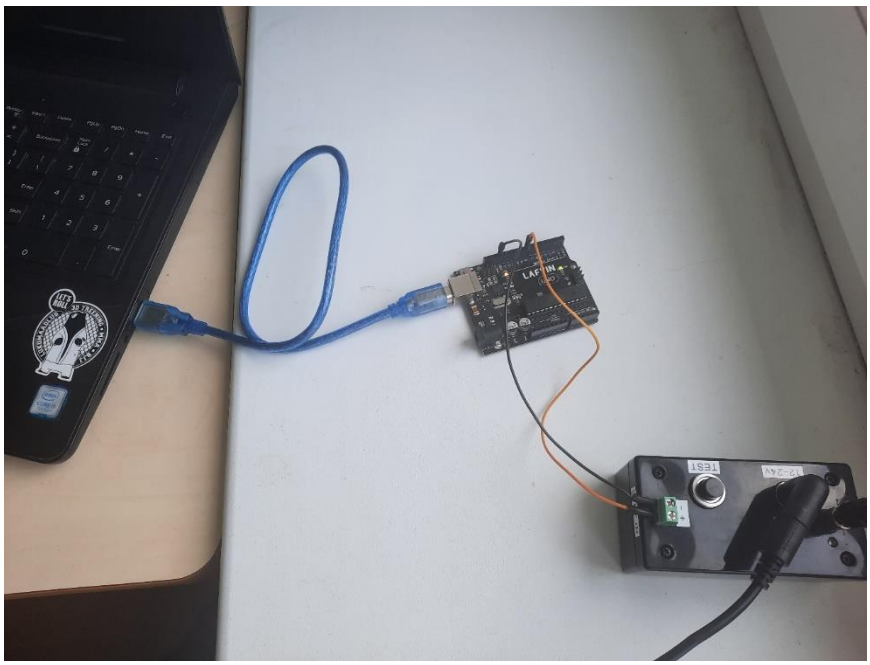

Figure 3. 7 Arduino Uno connected to EMP coil

The Arduino UNO board, connected to the computer and the EMP coil as shown in figure 3.7, is used here to trigger the EMP pulse when the camera starts recording. This is setup using a Python script. The board itself is loaded with the code, using the Arduino IDE, as shown in the methods section of this thesis. The Arduino board is connected to the PC via a USB cable. Then pin number '8' of the Arduino digital output, according to the Python script, is connected to the positive (+) sync of the EMP coil's connector terminal (orange wire in the figure below). The ground pin in the Arduino board is connected to the negative terminal of the power/ button box of the EMP coil (black wire in Figure 3.7)

When the Python script activates pin '8', the EMP signal, as explained previously, is generated an impulse by the EMP coil. This part of the setup is only used during the start of the test or experiment and can be removed once the EMP signal has been generated. Since the code to setup the board for this task is stored in the board memory, the Arduino IDE needs to be used only once.

3.5 Python

Python programming language has been chosen as main programming tool. This, due to its simple use and availability of built libraries to simplify many tasks. The interpreter engine used is Python version 3.8 and is run on a Windows 10 computer. The development environment is PyCharm and the Python libraries used are given below.

Table 3. 5 Python libraries used

Python Library	Version	Purpose
pyueye	4.90.0.0	The python library of the Ueye camera. Camera functionalities can be accessed via this library
opencv	4.5.1.48	To display, resize and save images
Datetime	Inbuilt library	To import computer clock time and to help timestamping
pandas	0.0.97	To save image frame names with corresponding timestamps as a csv file.
serial	3.5	To enable communication with Arduino board
time	Inbuilt library	To allow for sleep time in the programme

4. METHOD

4.1 Image acquisition

Image acquisition is one of the most crucial parts in this thesis. The time at which each image is obtained must be recorded as accurately as possible to ensure good synchronization with the data obtained from the IMU. The computer time is used in this thesis to timestamp the images when received from the camera.

The uEye camera used in this thesis comes along with a dedicated software application called "uEye Cockpit" [24]. Camera functionalities as Exposure settings, frame rates, timestamping of video and saving can be accessed via this application. The user interface is shown in figure 4.1 .

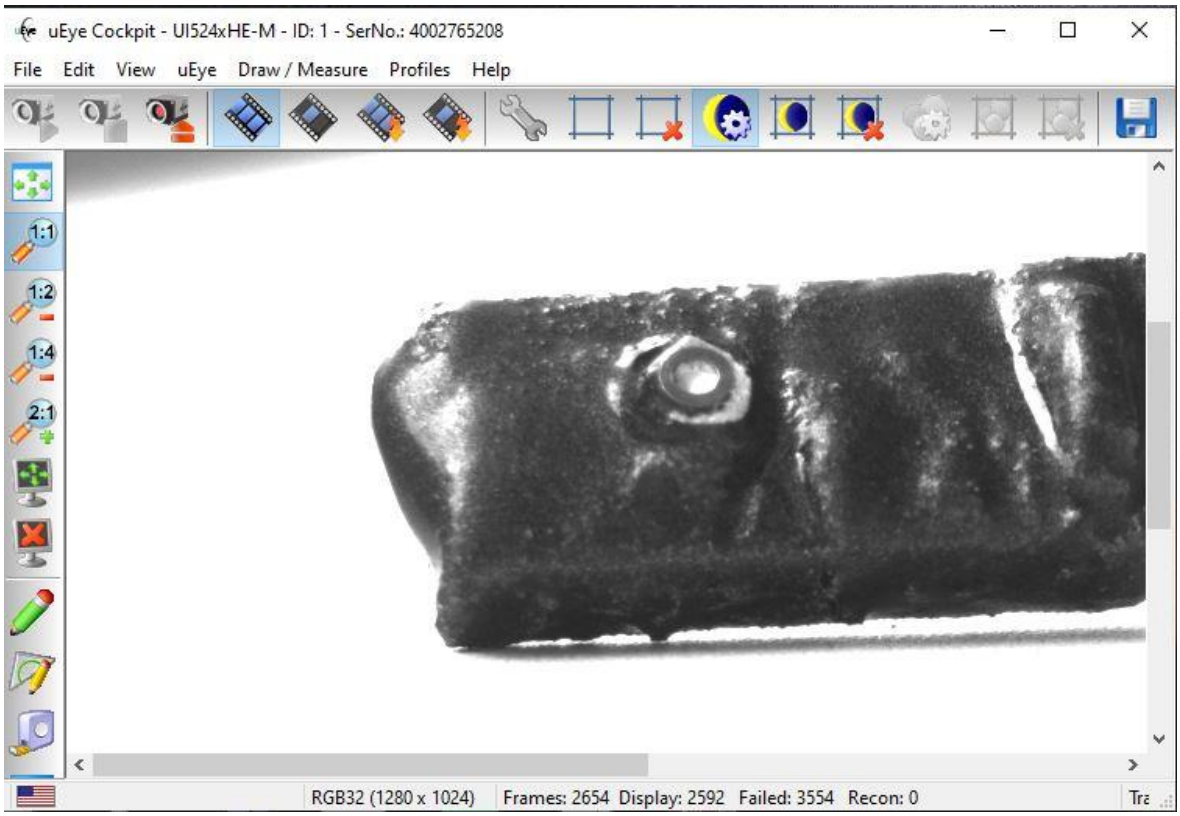

Figure 4. 1 uEye Cockpit user interface

Unfortunately, this software does not allow for the flexibility that is required to fulfil the objectives of this thesis. uEye has a Python library (pyueye) which can be used in a Python script to access various functionalities of the camera. Unfortunately, the library does not allow for continuous capture of images via triggering by a Python script and

also doesn’t have an option to timestamp images. Therefore, a different approach was needed to obtain single images with accurate timestamp.

OpenCV is a Python library that is mainly used for computer vision or machine vision applications. This library was used in this thesis to obtain the video (image frames) from the camera and display it in a separate window. One of the advantages of using the OpenCV library is that this displayed video can also be cut down into individual image frames and be saved to the computer. The Python script to setup the camera to record video and display the images using the OpenCV library was already available in the IDS Imaging Development Systems GmbH website. Therefore, the main modification to be done to the script was to obtain the individual frames from the displayed window, name each image frame sequentially, add timestamps (time at which the images were displayed on a window) , save the images and create a file which lists the image names along with their corresponding timestamps.

The uEye library, pyueye, does not include a function to save the images to the computer. Therefore, a while loop was used to continuously read from the camera memory, display and save the images to the computer. This presented several challenges. Mainly, since the images couldn’t be saved to the computer directly from the camera memory, they had to be saved using OpenCv functions after being displayed in a separate window. This created additional delays and sometimes even duplicated several images as the communication speed between the camera and the computer varied.

These unpredictable delays made the timestamping unreliable. Also, since this was an industrial camera, there weren’t any other available methods to save the images directly from the camera. So, after several weeks of trying to accomplish the above task, it was decided to abandon the uEye camera and to proceed with a simpler camera which would provide much more accessibility.

The web camera chosen to replace the uEye camera was the Canyon C3. This camera can capture images at a resolution of 720p and is connected to the computer via a USB cable. The Canyon C3 camera doesn’t include a software application but it can be controlled easily via a Python script to trigger snapshots, display images and also save them to the computer. The time at which the trigger signal is sent was also recorded and was added as the timestamp for that particular image frame. The timestamp, obtained from the computer clock, were added to a csv file with the corresponding image frame name.

4.2 Data Acquisition

The wearable sensing unit records several types of data, as showed in Figure 4.2: the time elapsed since turning on the sensor, battery level, pressure, temperature and IMU data (acceleration in 3 axis, gyroscopic measurements in 3 axis, magnetic field strength in 3 axis).

These data are saved to a text file and are separated by commas. This can be imported to an Excel file after the experiment to perform analysis or visualize variation in data by the use of graphs. One such test example is shown below in figure 4.2 .

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Data No.	Time	Battery	Pressu	Tempera	Acce. X	Acce. Y	Acce. Z	Gyro X	Gyro Y	Gyro Z	Mag X	Mag Y	Mag Z
2	1	1449	4.07	1010.69	29.7	0.3484	0.1437	10.1897	0.1	0	-0.6	-14.1	-153.3	-72.9
3	2	1459	4.07	1010.47	29.7	0.3735	0.1425	10.1921	0.2	0	-0.7	2.7	-155.7	-69.9
4	3	1469	4.07	1010.67	29.7	0.3627	0.1125	10.1945	0.2	-0.1	-0.7	0.3	-160.5	-65.7
5	4	1479	4.07	1010.27	29.7	0.3591	0.1245	10.2256	0.3	-0.1	-0.6	5.1	-170.1	-69.9
6	5	1489	4.07	1010.25	29.7	0.3472	0.1281	10.2244	0.2	0	-0.7	-4.5	-153.3	-68.1
7	6	1499	4.07	1010.32	29.7	0.3639	0.1161	10.1789	0.2	0	-0.6	-14.1	-160.5	-68.7
8	7	1509	4.07	1010.29	29.7	0.3771	0.1365	10.2148	0.2	-0.1	-0.7	17.1	-153.3	-72.3
9	8	1519	4.07	1010.38	29.7	0.3627	0.1472	10.1993	0.2	0	-0.6	12.3	-167.7	-66.9
10	9	1529	4.07	1010.27	29.7	0.3603	0.1281	10.2208	0.2	0	-0.6	-6.9	-158.1	-69.9
11	10	1539	4.07	1010.27	29.7	0.3615	0.1281	10.216	0.2	-0.1	-0.6	0.3	-162.9	-71.1
12	11	1549	4.07	1010.09	29.7	0.3639	0.1233	10.216	0.2	-0.1	-0.7	7.5	-155.7	-66.3
13	12	1559	4.07	1009.99	29.7	0.3603	0.1161	10.2053	0.2	-0.1	-0.7	9.9	-158.1	-57.9
14	13	1569	4.07	1010.06	29.7	0.3615	0.1353	10.21	0.2	0	-0.7	-11.7	-162.9	-62.7
15	14	1579	4.07	1010.05	29.7	0.3543	0.1197	10.1705	0.2	0	-0.6	-2.1	-153.3	-73.5
16	15	1589	4.07	1010.06	29.7	0.3567	0.1269	10.1945	0.1	0	-0.7	-18.9	-160.5	-60.9
17	16	1599	4.07	1010.03	29.7	0.3543	0.1317	10.1945	0.2	0	-0.7	-11.7	-148.5	-69.3
18	17	1609	4.07	1010.03	29.7	0.3651	0.1353	10.1921	0.2	-0.1	-0.6	17.1	-162.9	-71.7

Figure 4. 2 Wearable sensor unit data displayed in an Excel sheet

The sensing unit is turned on when a magnet is placed near it. Then it starts recording data every 10 milliseconds (100Hz). The data recording time can be changed by modifying the "time" text file available in the device memory. Once elapsed time reaches the set value in the text file, the sensing unit stops recording data and turns itself off. The sensing unit is then connected to the computer via a USB cable and the recorded data can be extracted from the device memory card.

To synchronize the data from the sensing unit to camera image frames, an electromagnetic pulse coil is used. The pulse is triggered by the Python script which sends signal to an Arduino UNO board output pin which in turn sends a 5V signal to the EMP coil controller. The pulse generated rapidly spike, in both positive and negative directions depending on the orientation and placement of the sensing unit within the coil, the geomagnetic readings obtained by the sensing unit. Since the trigger signal is

sent to the coil at the same time at which the signal is sent to the camera, delays are minimal and matching the image frames with sensor readings becomes easier.

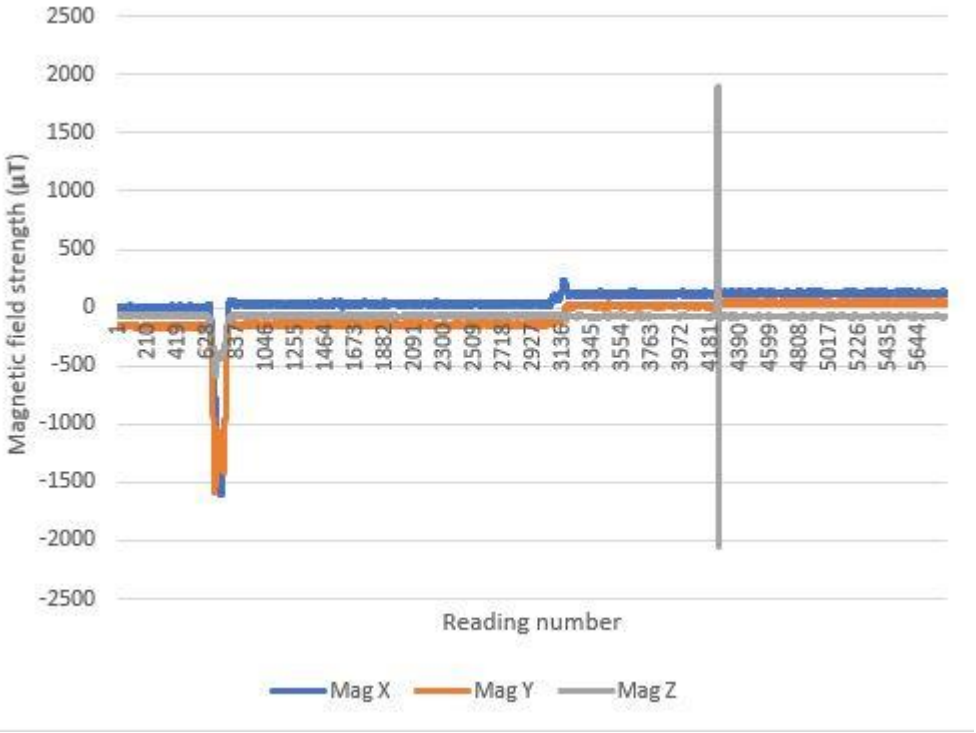

Figure 4. 3 Spike in Magnetic Strength reading (Mag Z)

The figure 4.3 shows the magnetic strength readings of all three-axis obtained by the sensing unit. The x axis is the number of readings and the y axis is the magnetic field strength recorded. The spike in the readings (Mag Z value) is clearly visible near reading no 4181. By equating the start point of the spike to the time at which the signal was sent to the coil, consequent timestamps of recorded data can be obtained.

4.3 Python Script

The main Python script used in this thesis focuses on the camera image frame acquisition and timestamping of images. It also includes a set of code for the Arduino UNO unit which would send a signal to the EMP coil at the same time the image capture begins. These codes, include establishing communication with the Arduino board and also a function that would turn on a particular output pin in the board to output a voltage of 5V. The code used here and elsewhere in this thesis are available in a repository [25] (https://github.com/BalakrishnanGuruprasath/cameraimusync) .The code flow is shown in the flowchart shown in figure 4.4 .

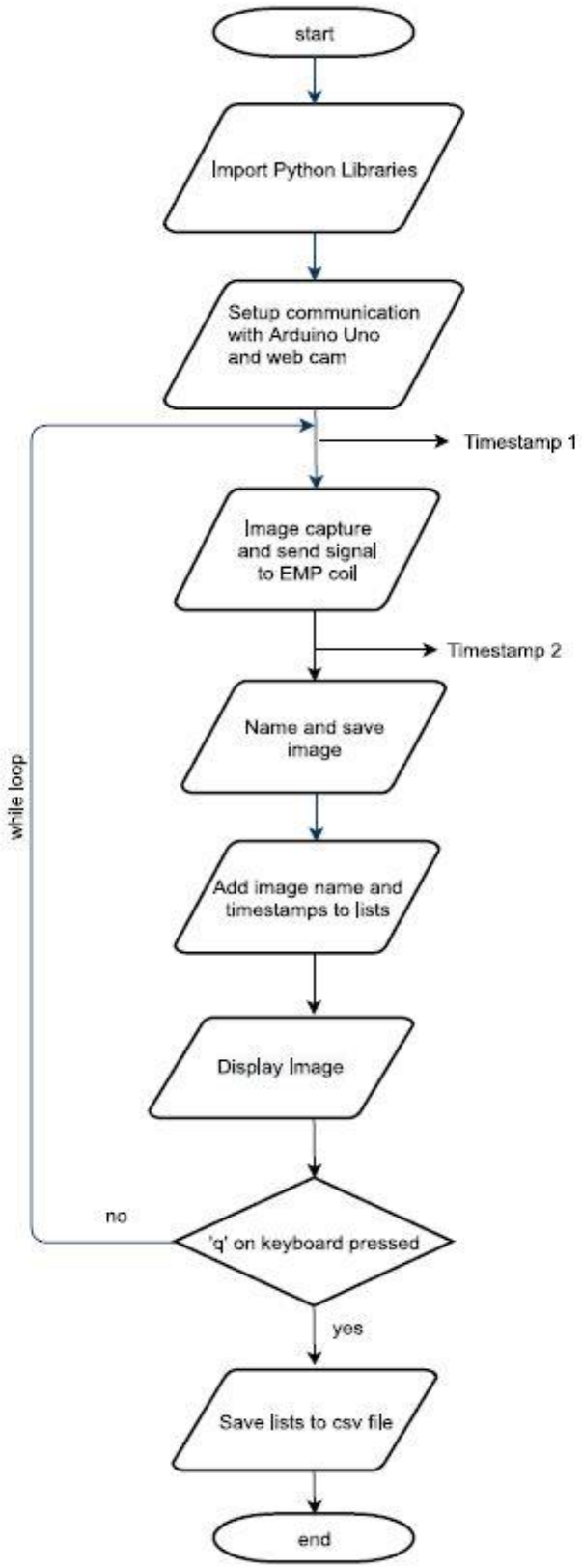

Figure 4. 4 Image capture and EMP coil activation script flowchart

Since the initial camera used was the uEye camera, the first version of the image acquisition Python code involved a lot of camera and communication setup. As mentioned before, this part of the code was directly obtained from the uEye camera manufacturer's website. But, after replacing the uEye camera with the web camera, the setup code was no longer needed and was replaced by image capture (trigger) code using the Python OpenCv library. The trigger for image capture code was preceded and followed by a couple of variables which recorded the computer time. This was done to obtain the trigger time as accurately as possible.

After step 3 (image capture), the image was displayed in a separate window and the image is saved to a specified folder in the computer. Then the image name along with the two timestamps are added to separate Python lists. These lists were added to a csv file after the completion of the test. Since steps 3 to 6 were placed inside a while loop, after step 6 (display image), the operation of code returned back to step 3. This continued until the key 'q' was pressed which breaks the loop and finishes the program after saving the lists to a csv file.

4.4 Arduino code

The Arduino Uno board used in this thesis was step up with a simple code to output 5V out of output pin number '8' when instructed by the main Python script. If the serial communication with Python script was available and the data sent from it was '1' then the output from pin '8' would be high (5v). If not, then the output would be low (0v). Although the function to activate the output pin in the Python script is within the while loop, it is run only once (due to implementation of an 'if' statement).

```
void loop(){

  if(Serial.available() > 0){
    serialData  = Serial.read();
    Serial.print(serialData);

    if(serialData == '1'){
      digitalWrite(pin, HIGH);}
    else if(serialData == '0'){
      digitalWrite(pin, LOW); }
  }
```

Figure 4. 5 Code written into the Arduino Board using Arduino IDE

5. TESTS AND RESULTS

5.1 Image capture delays

The first task in this thesis was to approximate the difference between image timestamps to the actual time at which the image was captured. This difference would approximately be equal to the camera exposure delay and other minor delays associated with communication between the PC (Python script) and the web camera. The transmission delay from the web camera to the computer can be neglected in this thesis because the timestamps are not logged when the image is received but when the command is sent out to the camera to capture the image. Also, the frame rate at which the camera captures images is not significant as each individual image frame is captured via a trigger command in the Python script.

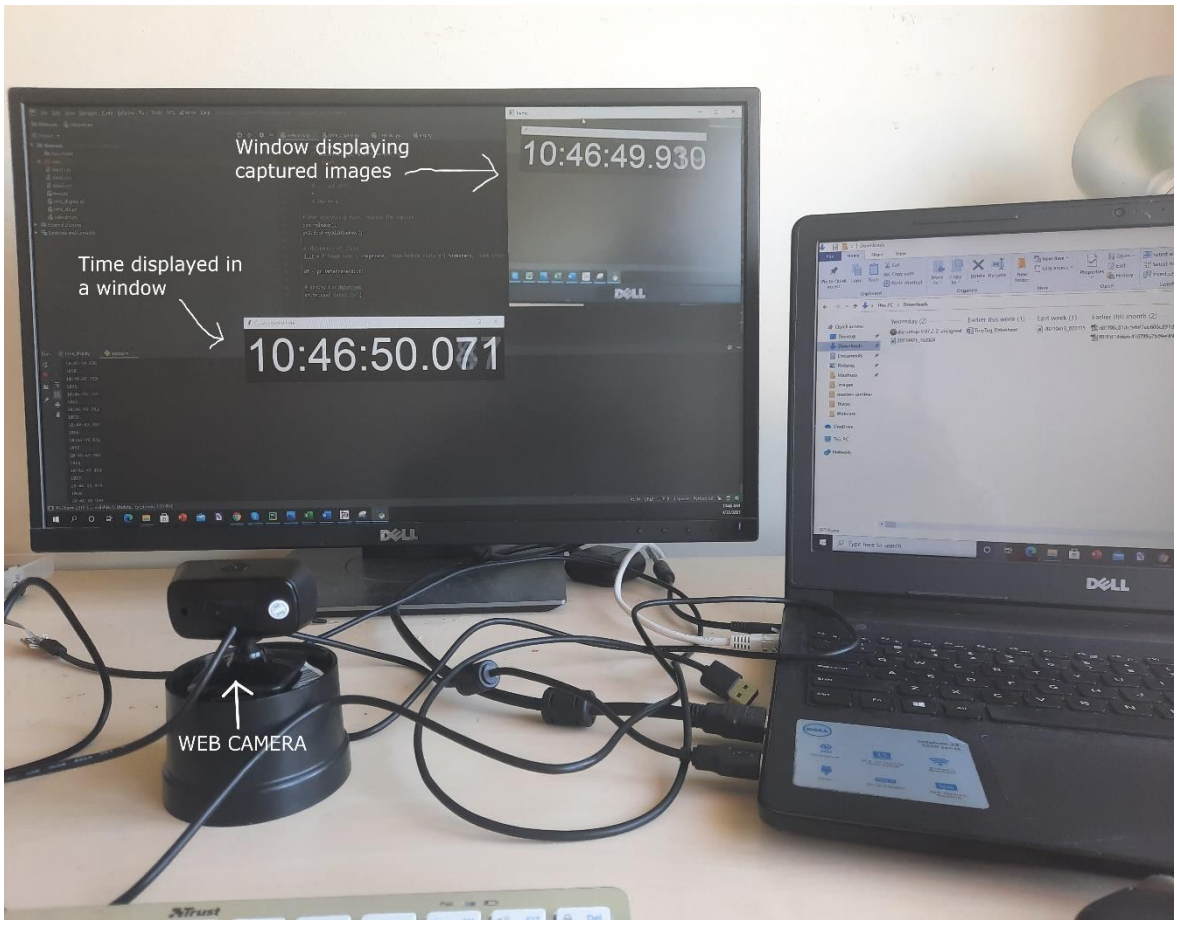

Figure 5. 1 Delay approximation setup

The test setup of Figure 5.1 is explained hereafter. The image timestamps, according to the Python script, was obtained from the computer system clock. This time was given

up to milliseconds and in the format of "HH:mm:ss.000", and displayed on screen in real time via a different Python script run simultaneously. The main image capture script, explained in the previous section, without the synchronization EMP signal setup was then run. The camera was pointed to the screen displaying the system time. The time captured by the web camera was compared to the system clock timestamps saved to the csv file. The difference in these times gave the approximate delays. The differences were then added to the timestamps to make it as accurate as possible. The current time window is shown in figure 5.2.

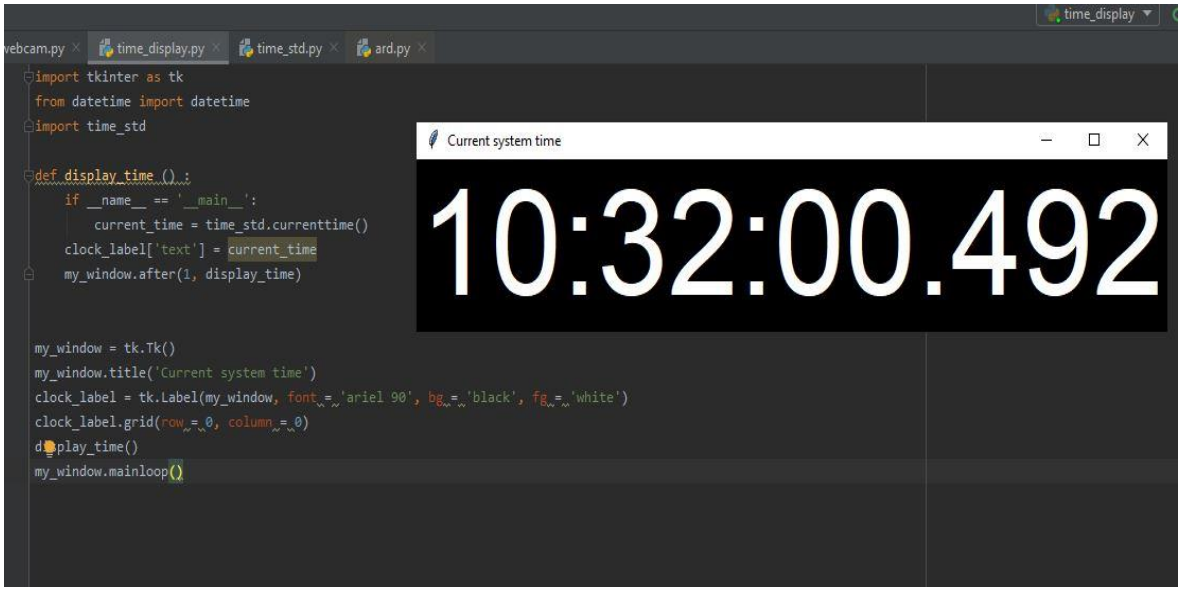

Figure 5. 2 System time displayed in real time

A Python library called "TKinter" was used to display the computer system time. The current system time was obtained using the inbuilt datetime Python library. The script for this was saved in a different Python file and was accessed by both the image capture script and also the time display script.

Once this test was run for a couple of minutes, the test was stopped and the images saved were analysed with the timestamps. The csv file was opened in Microsoft excel and the timestamp columns' format was changed to the format which was used to display the system time. Then several images containing the time were compared to the timestamp (time just before capture). This was done manually to find out the approximate delay between the initiation of trigger signal by the Python script and the time at which the image was eventually captured.

For example, in this test 775 images were captured and saved. The corresponding timestamps were also saved to a csv file. Image 'ex322' displays a time of 13:50:42.932

and the timestamp, the one right before capture, corresponding to that image in the csv file logged in a time of 13:50:42.858. This shows a difference of 74 milliseconds. The following figure 5.3 shows image ‘ex322’ and the time captured by it.

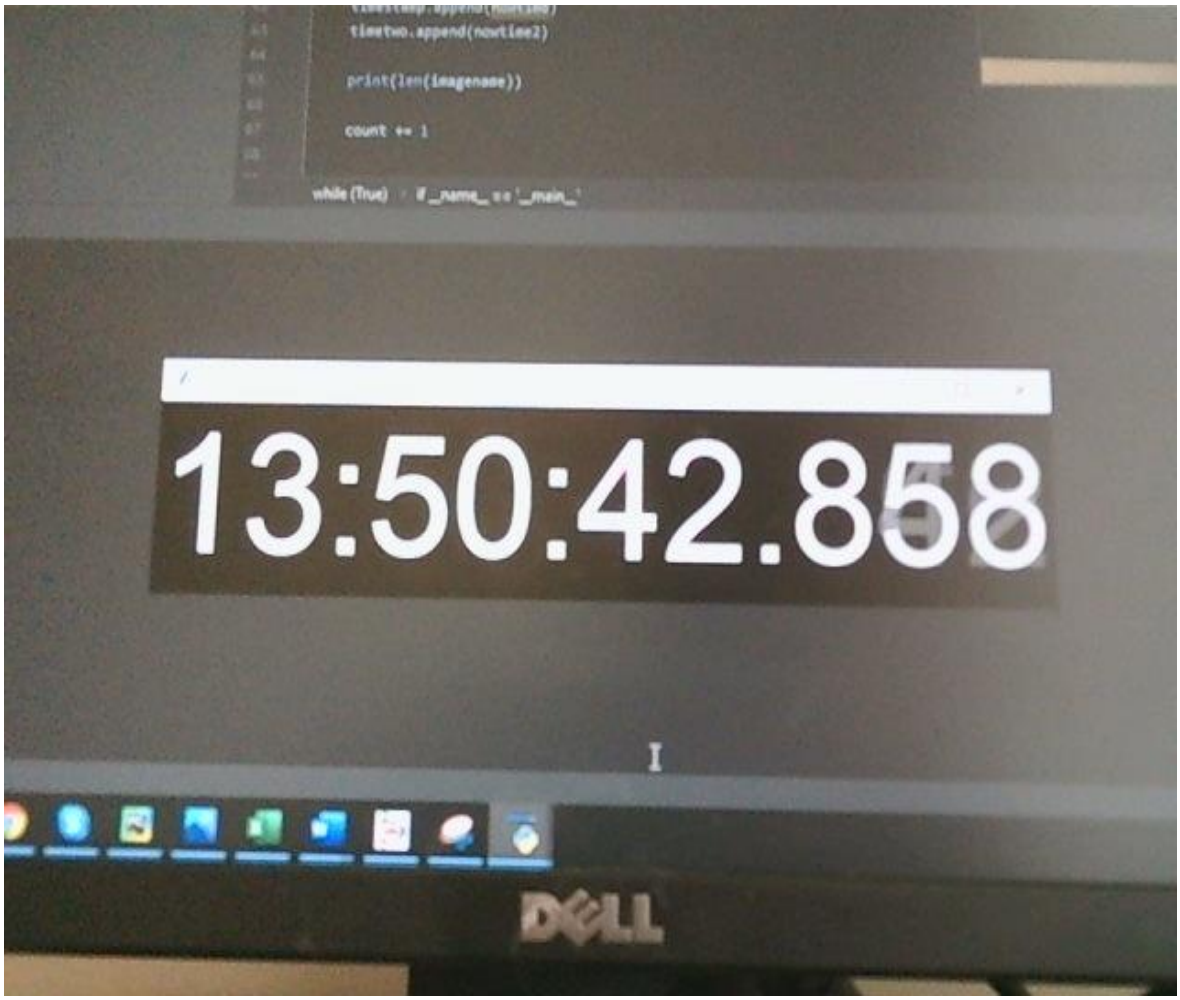

Figure 5. 3 Image ‘ex322’

Unfortunately, since the camera itself was not fast enough (capture rate a maximum of 30 frames per second), sometimes the millisecond region of the image was unclear. In the case shown above, it was reasonably clear and therefore was used to calculate the time difference. The figure 5.4 below shows the difference in time of some of these images. Some cells have been highlighted in red because their captured images are not clear enough to deduce the time.

1		Image name	Time before capture	Time after capture		Time in image	Difference
322	320	ex320	13:50:42.833	13:50:42.886			
323	321	ex321	13:50:42.901	13:50:42.917			
324	322	ex322	13:50:42.932	13:50:42.949		13:50:42.858	0:00:00.074
325	323	ex323	13:50:42.965	13:50:42.981		13:50:42.877	0:00:00.088
326	324	ex324	13:50:42.995	13:50:43.013		13:50:42.914	0:00:00.081
327	325	ex325	13:50:43.025	13:50:43.045		13:50:42.955	0:00:00.070
328	326	ex326	13:50:43.057	13:50:43.061			
329	327	ex327	13:50:43.073	13:50:43.093			
330	328	ex328	13:50:43.106	13:50:43.158		13:50:43.075	0:00:00.031

Figure 5. 4 Image timestamps along with the difference

As mentioned previously, the 'Time in Image' column values were added manually and the time difference, against 'Time before capture', was calculated. The difference in values varied from 40 millisecond up to 90 milliseconds (from random images within this test). This, obviously, also depends on the other processes running in the computer background. There were several user programs where running in the background while these data were collected. When many of the user programs were not operational, the difference in varied between 40 and 60 milliseconds. Comparing the delay times from various such tests it was safe to assume that the difference in times never exceeded 100 milliseconds.

5.2 Drop Test

The purpose of this test was to check if the images from the web camera can synchronize with the data from the wearable sensor unit. For this initial test, the wearable sensor is attached to a small box and is dropped from a height. In this way, the acceleration values of a particular axis (the z axis in this case) will change from the gravitational acceleration value (around 10m/s-2), just before releasing the box, back to a similar value as before when it comes to rest on the floor. This makes it easier to verify the synchronization of the images with the sensor data. The way the sensor unit was attached to the box is shown the figure 5.5.

Figure 5. 5 Sensor unit attached to a box for test

The sensor unit was programmed to collect data for 1 minute after being turned on and then turned itself off. The sensor unit was turned on by a magnet and was placed inside the EMP coil loop and the image acquisition Python script was run. The command for synchronization EMP signal was sent to the coil by the Python script at the same time the image stream started. But, tests with the web camera pointing towards the leds attached to the EMP coil circuit box showed that there was around 0.5 second delay from when the command was run in the Python script and the time at which the EMP coil was activated. This delay is clearly shown in the figure 5.6. Here, 'ex0' is the first image captured and 'ex1' is the second image captured. The led is off in the first image and is only turned on in the second image (ex1). Therefore, for this test and all the tests

that follow, the first image is disregarded. Images 'ex0' and 'ex1' are compared in the figure 5.7.

	A	B	C	D
1		Image name	Time before capture	Time after capture
2	0	ex0	19:49:13.515	19:49:13.950
3	1	ex1	19:49:13.998	19:49:13.999
4	2	ex2	19:49:14.016	19:49:14.018
5	3	ex3	19:49:14.034	19:49:14.049
6	4	ex4	19:49:14.066	19:49:14.077
7	5	ex5	19:49:14.089	19:49:14.108
8	6	ex6	19:49:14.124	19:49:14.140
9	7	ex7	19:49:14.155	19:49:14.172

Figure 5. 6 Time difference between the 1st and 2nd images

Figure 5. 7 Image 'ex0' and the left and image 'ex1' on the right

Next, the box was carried over in front of the web camera and was held at a height from the floor for a few seconds. This was done to bring the acceleration values as close to environmental values before release. Then, the box was dropped and image capture and data collection continued until the sensor unit turned off automatically. The data from the sensor unit was then transferred to the computer for analysis. In this test, a total of 1732 images were captured. The sensor data count was 5849. Since the data were saved into a csv text file, they were transferred to an Excel sheet where they can be analysed properly.

The first step during analysis was to identify the point at which the EMP signal was recorded by the sensor unit. Since amount of data obtained by the sensor unit, as

mentioned before, is very large, it was easier to plot the 'Data no.' column of the sheet vs the magnetic field strength columns (Mag X, Mag Y and Mag Z) to find out the peak point (the point at which the magnetic strength reading starts to spike). This plot is shown in figure 5.8.

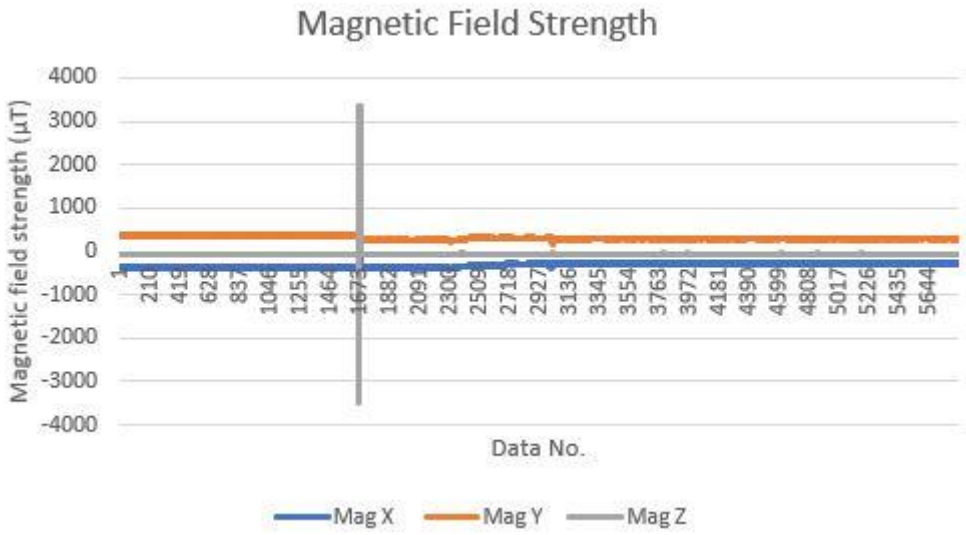

Figure 5. 8 Magnetic field strength plot

The above figure shows that the electromagnetic pulse was recorded by the sensor unit at approximately Data no. 1673. The data around this point, on the Excel sheet, were checked to pinpoint the exact point at which the pulse started to register. The exact point was Data no. 1672 with a Mag Z value of – 3510.9 µT (highlighted in figure 5.9). This peak lasts for 4 more sets (1673 to 1676) and then flips directions for 5 more sets (1677 to 1681). Data no. 1671 had a value of -74.1 µT and data no. -72.9 had a value of -68.4 µT. This shows the clear difference from when the pulse was active and when it was not.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Data No	Time	Battery	Pressure	Tempe	Acc. X	Acc. Y	Acc.Z	Gyro X	Gyro Y	Gyro Z	Mag X	Mag Y	Mag Z
1668	1667	18117	4.01	1012.12	24.48	0.3531	0.0048	10.1322	0.3	0	-0.7	-369.3	369.9	-72.9
1669	1668	18127	4.01	1012.07	24.48	0.3687	0.0215	10.1478	0.3	0	-0.7	-371.7	372.3	-78.3
1670	1669	18137	4.01	1012.07	24.48	0.3675	-0.0024	10.1191	0.3	-0.1	-0.6	-366.9	374.7	-80.7
1671	1670	18147	4.01	1012.12	24.48	0.3711	0.0108	10.1239	0.3	0	-0.7	-359.7	379.5	-77.7
1672	1671	18157	4.01	1012.2	24.48	0.3651	-0.0048	10.1215	0.4	0.1	-0.7	-374.1	377.1	-74.1
1673	1672	18167	4	1012.03	24.48	0.3711	0	10.1011	0.3	0	-0.7	-2183.7	-198.9	-3510.9
1674	1673	18177	4	1012.13	24.48	0.3855	0.0144	10.137	0.3	0	-0.7	-2171.7	-198.9	-3497.1
1675	1674	18187	4.01	1012.21	24.48	0.3759	-0.0084	10.1203	0.3	0	-0.6	-2188.5	-198.9	-3503.1
1676	1675	18197	4.01	1012.18	24.48	0.3711	0.0132	10.0867	0.4	0.1	-0.6	-2190.9	-203.7	-3509.1
1677	1676	18207	4.01	1012.14	24.48	0.3627	0.0048	10.149	0.4	0	-0.7	-2193.3	-189.3	-3506.1
1678	1677	18217	4.01	1012.22	24.48	0.3771	-0.0096	10.1574	0.4	0	-0.7	1452.3	852.3	3353.1
1679	1678	18227	4	1012.02	24.48	0.3795	0.0084	10.1251	0.4	0	-0.6	1469.1	852.3	3355.5
1680	1679	18237	4	1012.07	24.48	0.3735	-0.0227	10.1418	0.4	0	-0.6	1454.7	842.7	3366.9
1681	1680	18247	4.01	1012.11	24.48	0.3699	0.0048	10.143	0.2	0	-0.6	1469.1	840.3	3361.5
1682	1681	18257	4.01	1012.13	24.48	0.3771	0	10.1274	0.2	0.1	-0.7	1471.5	833.1	3362.7
1683	1682	18267	4.01	1012.24	24.48	0.3735	0.0096	10.1562	0.3	0.1	-0.6	-371.7	276.3	-72.9
1684	1683	18277	4.01	1012.07	24.48	0.3795	0.0156	10.1107	0.3	0	-0.6	-376.5	266.7	-74.7

Figure 5. 9 EMP signal register point (Data no. 1672)

Next, the timestamp of the second image captured (ex1) is added as the timestamp for data no. 1672 and the timestamps for the rest of the data are obtained by adding 10ms to the timestamp from the previous row. This was done because the sensor unit records data at a rate of 100 hz (every 10ms). The timestamp of 'ex1' of this test was 19:49:13.998. This and the consequent timestamps were added as shown in the figure 5.10 .

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Data No	Time	Battery	Pressure	Tempe	Acc. X	Acc. Y	Acc.Z	Gyro X	Gyro Y	Gyro Z	Mag X	Mag Y	Mag Z		Timestamp
1671	1670	18147	4.01	1012.12	24.48	0.3711	0.0108	10.1239	0.3	0	-0.7	-359.7	379.5	-77.7		
1672	1671	18157	4.01	1012.2	24.48	0.3651	-0.0048	10.1215	0.4	0.1	-0.7	-374.1	377.1	-74.1		
1673	1672	18167	4	1012.03	24.48	0.3711	0	10.1011	0.3	0	-0.7	-2183.7	-198.9	-3510.9		19:49:13.998
1674	1673	18177	4	1012.13	24.48	0.3855	0.0144	10.137	0.3	0	-0.7	-2171.7	-198.9	-3497.1		19:49:14.008
1675	1674	18187	4.01	1012.21	24.48	0.3759	-0.0084	10.1203	0.3	0	-0.6	-2188.5	-198.9	-3503.1		19:49:14.018
1676	1675	18197	4.01	1012.18	24.48	0.3711	0.0132	10.0867	0.4	0.1	-0.6	-2190.9	-203.7	-3509.1		19:49:14.028
1677	1676	18207	4.01	1012.14	24.48	0.3627	0.0048	10.149	0.4	0	-0.7	-2193.3	-189.3	-3506.1		19:49:14.038
1678	1677	18217	4.01	1012.22	24.48	0.3771	-0.0096	10.1574	0.4	0	-0.7	1452.3	852.3	3353.1		19:49:14.048
1679	1678	18227	4	1012.02	24.48	0.3795	0.0084	10.1251	0.4	0	-0.6	1469.1	852.3	3355.5		19:49:14.058
1680	1679	18237	4	1012.07	24.48	0.3735	-0.0227	10.1418	0.4	0	-0.6	1454.7	842.7	3366.9		19:49:14.068

Figure 5. 10 Timestamps added to the sensor unit data

The next task was to see if the images and the sensor data timestamps correlate with each other. To do this, the exact moment at which the box was released from a height must be known. The acceleration values recorded show the point at which the box was released. Again, similar to finding the magnetic field strength spike, the data numbers are plotted against the acceleration columns in the sensor data Excel sheet. Figure 5.11 shows the plot.

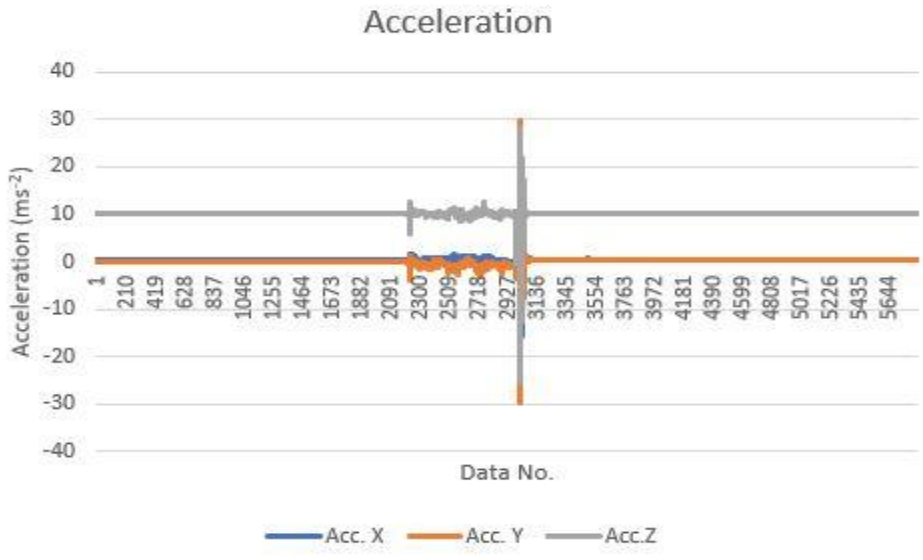

Figure 5. 11 Data no vs acceleration plot

The plot shows that the spike in acceleration values occurred somewhere around data no 2927 and 3136. Looking through the data it is clear that the Acceleration in the Z axis has a value of around 10 ms-2 at rest (approximate gravitational acceleration) and therefore is the axis that should be focused upon. It can be seen from figure 5.12 that the acceleration values in the z axis start decreasing from the rest value at data no. 2983. The timestamp at this point is 19:49:27.108 (highlighted in blue in figure 5.12). The values keep changing until data no 3044 where the value reaches 10.55 and then goes back to the standard rest value. This shows that the box came to rest at this point. The timestamp here is 19:49:27.718 (highlighted in orange in figure 5.13).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Data No	Time	Battery	Pressure	Tempe	Acc. X	Acc. Y	Acc.Z	Gyro X	Gyro Y	Gyro Z	Mag X	Mag Y	Mag Z		Timestamp
2981	2980	31247	4	1012.44	24.84	-0.085	-0.8607	10.5033	-2.8	-13.3	3.7	-278.1	297.9	-59.1		19:49:27.078
2982	2981	31257	4	1012.53	24.84	-0.0132	-0.905	10.6314	-1.9	-8.9	2.4	-299.7	307.5	-63.9		19:49:27.088
2983	2982	31267	4	1012.6	24.84	-0.2945	-0.905	10.4399	1.3	-3	2.1	-292.5	319.5	-56.1		19:49:27.098
2984	2983	31277	4	1012.41	24.84	-0.3567	-0.8595	10.2053	4.6	-0.1	3.1	-287.7	307.5	-61.5		19:49:27.108
2985	2984	31287	4	1012.57	24.84	-0.2227	-0.7817	9.9778	5.5	2.4	3.2	-294.9	305.1	-62.1		19:49:27.118
2986	2985	31297	4	1012.46	24.84	-0.2143	-0.7422	9.5504	3	5.7	1.9	-268.5	295.5	-64.5		19:49:27.128
2987	2986	31307	4	1012.45	24.84	-0.1544	-0.7218	9.007	-1.9	8.6	0.4	-287.7	302.7	-69.3		19:49:27.138
2988	2987	31317	3.99	1012.54	24.84	-0.2705	-0.9984	8.351	-8.4	11.8	-0.2	-282.9	305.1	-71.7		19:49:27.148
2989	2988	31327	4	1012.6	24.84	-0.7314	-0.984	7.5285	-16.3	13.2	2.7	-294.9	309.9	-63.9		19:49:27.158
2990	2989	31337	4	1012.46	24.84	-0.893	-1.002	6.5709	-27.9	6.4	8.2	-266.1	302.7	-72.3		19:49:27.168
2991	2990	31347	4	1012.4	24.84	0.3292	-0.9481	5.8526	-39.8	-6.8	12.6	-285.3	321.9	-69.9		19:49:27.178
2992	2991	31357	4	1012.38	24.84	-2.4457	-1.3791	2.5522	-46.1	-8.2	15.7	-285.3	314.7	-63.9		19:49:27.188

Figure 5. 12 The point at which the acceleration values of Z axis start reducing

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Data No	Time	Battery	Pressure	Tempe	Acc. X	Acc. Y	Acc.Z	Gyro X	Gyro Y	Gyro Z	Mag X	Mag Y	Mag Z		Timestamp
3038	3037	31817	4	1012.35	24.86	0.5854	-10.8242	7.5453	-199.2	-118.7	23.7	-256.5	276.3	-57.9		19:49:27.648
3039	3038	31827	4	1012.52	24.86	14.601	-4.203	17.0993	-59.8	-146.7	17.1	-282.9	297.9	-63.3		19:49:27.658
3040	3039	31837	4.01	1012.4	24.86	-1.2905	0.1185	2.8479	8.9	38.7	7.6	-275.7	295.5	-60.3		19:49:27.668
3041	3040	31847	4.01	1012.5	24.86	7.0653	-7.3514	8.5341	5.7	-67.8	6.6	-285.3	300.3	-62.7		19:49:27.678
3042	3041	31857	4	1012.57	24.86	3.9959	1.3934	10.8529	29	-6.4	2.8	-297.3	297.9	-62.7		19:49:27.688
3043	3042	31867	4	1012.51	24.86	3.1699	-2.6695	8.479	32.2	43.3	-3.2	-299.7	295.5	-63.3		19:49:27.698
3044	3043	31877	4	1012.44	24.86	1.7873	0.6009	12.8341	21.8	43.9	-5.4	-270.9	288.3	-60.9		19:49:27.708
3045	3044	31887	4	1012.59	24.86	-1.5311	-0.1652	10.55	38.4	50.7	-6	-290.1	290.7	-62.1		19:49:27.718
3046	3045	31897	4	1012.53	24.86	-1.8495	2.1799	10.2651	42.1	29.1	-5.1	-287.7	273.9	-59.1		19:49:27.728
3047	3046	31907	4	1012.59	24.86	-1.9788	1.6005	8.418	31.1	10.4	-2.4	-275.7	283.5	-53.1		19:49:27.738
3048	3047	31917	4	1012.51	24.86	-0.2789	2.788	9.0309	15.1	-8.2	-0.6	-282.9	285.9	-59.7		19:49:27.748
3049	3048	31927	3.99	1012.52	24.86	1.0271	1.907	9.0141	-3.2	-11.2	0.5	-290.1	281.1	-57.3		19:49:27.758
3050	3049	31937	4	1012.64	24.86	1.1217	0.9373	10.161	-22.4	-5.1	0.4	-275.7	276.3	-59.7		19:49:27.768
3051	3050	31947	4	1012.68	24.88	0.7338	-0.7518	10.8026	-23.7	-0.4	-0.1	-287.7	285.9	-53.1		19:49:27.778

Figure 5. 13 The point at which the acceleration values of z axis reaches back to the nominal value

The final task was to check if the image frames at the point of release and at the point of rest correspond to the timestamps from the above data. The image frame just before release was ‘ex363’. The final rest on the floor occurs at image frame ‘ex381’. These images are shown in the figure 5.14. The timestamps of those image frames are illustrated in figure 5.15. The timestamps of these image frames have been compared to the timestamps of the corresponding sensor data in the table 5.1.

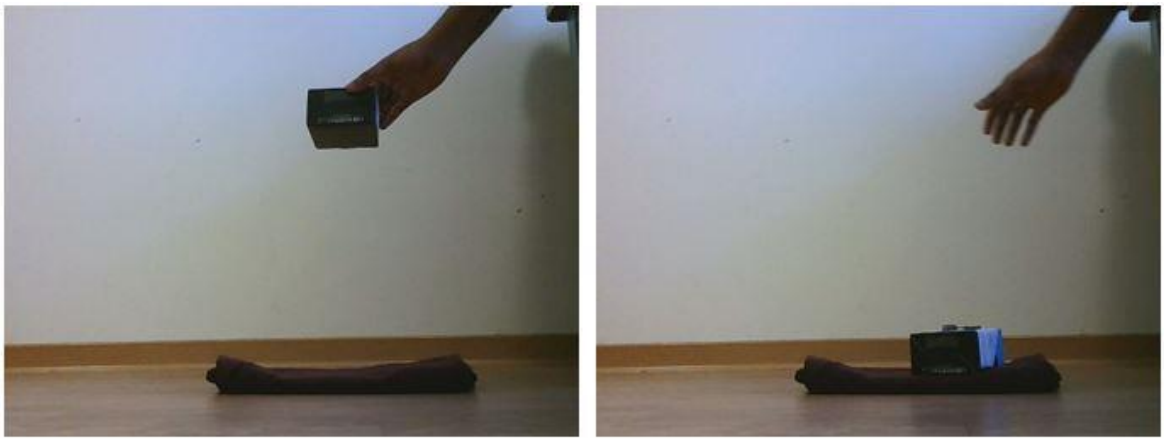

Figure 5. 14 Image ‘ex363’ on the left and image ‘ex381’ on the right

	A	B	C	
1	Image no	Image name	Time before capture (Timestamp 1)	Time (Tim
365	363	ex363	19:49:27.128	
366	364	ex364	19:49:27.172	
367	365	ex365	19:49:27.203	
368	366	ex366	19:49:27.234	
369	367	ex367	19:49:27.256	
370	368	ex368	19:49:27.288	
371	369	ex369	19:49:27.336	
372	370	ex370	19:49:27.373	
373	371	ex371	19:49:27.399	
374	372	ex372	19:49:27.431	
375	373	ex373	19:49:27.467	
376	374	ex374	19:49:27.500	
377	375	ex375	19:49:27.532	
378	376	ex376	19:49:27.560	
379	377	ex377	19:49:27.595	
380	378	ex378	19:49:27.625	
381	379	ex379	19:49:27.670	
382	380	ex380	19:49:27.704	
383	381	ex381	19:49:27.744	

Figure 5. 15 Image timestamps

Table 5. 1 Timestamps of sensor data and image frames compared

Position	**Timestamp of sensor data**	**Timestamp of image**	**Difference**
Release	19:49:27.108	19:49:27.128	20 milliseconds
Rest	19:49:27.718	19.49.27.744	16 milliseconds

Table 5.1 also shows the time difference between the timestamp of the sensor unit data and the timestamp of the image frames. The differences are very small and can be attributed to the lower frame rate of the web camera and other variable delays. This test also shows that the image frames are synchronized to a large extent with the sensor data. A GUI can then be created to display the image frames with their corresponding sensor data by using the timestamps as the common quantity. This can be further enhanced by using a camera (high speed camera) which records images at the same speed as the sensor unit.

5.3 Gait Cycle Test

In this test, two wearable sensor units are attached to a leg, one on the thigh and the other just below the calf, to perform gait analysis as detailed in [26]. The goal here was to synchronize the sensor data with the image frames obtained by the web camera. The top sensor (the one attached to the thigh) is attached at a distance of 28 cm from the hip and the bottom sensor is attached at a distance of 26.5 cm from the knee. The figure 5.16 shows the attachment of the sensors to the leg.

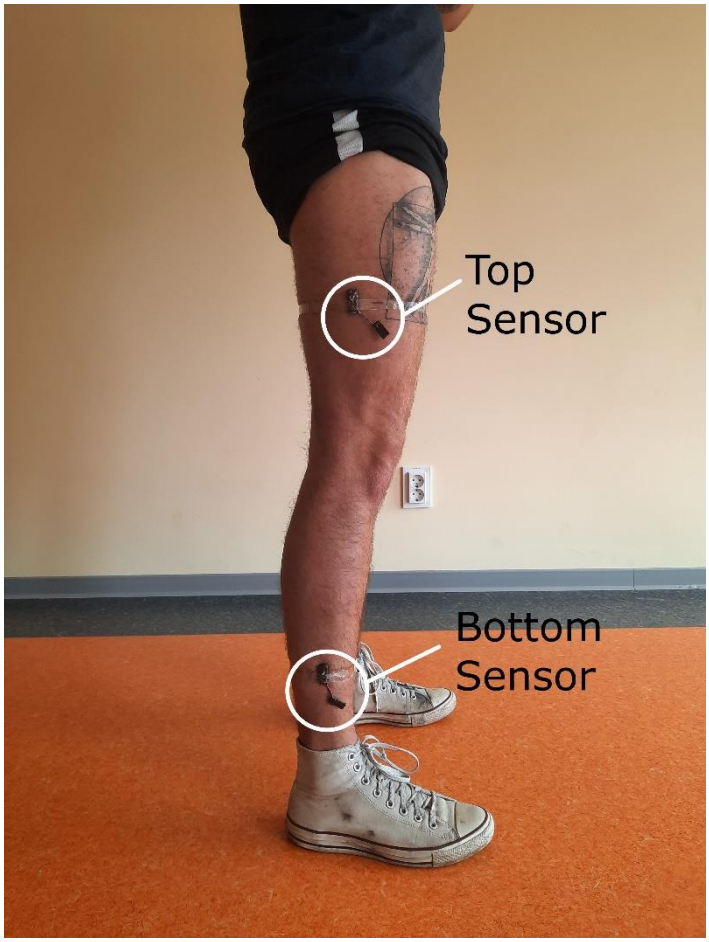

Figure 5. 16 Sensor units attached to the thigh and lower leg

Similar to the box drop test performed before, the sensors were turned on by a magnet and then the image acquisition script was run. The EMP coil was placed around the two sensors so that the synchronization electromagnetic pulse is sent to the sensors at the same time. Since the sensors were attached to the leg further apart than the diameter of the EMP coil, the leg was bent in the way shown in figure 5.17 to ensure the sensors fit within the coil.

Figure 5. 17 EMP synchronization coil setup

After the above-mentioned synchronization signal was sent, the subject walked to a position near the left edge of the web camera image frame. After, a few seconds of rest, the subject walked across the frame to complete 1 gait cycle. When the cycle was completed, the subject stood still until the sensors turned off automatically (1 minute after turning on the sensor) and then the camera was also turned off.

The analysis part of this test is very similar to that of the box drop test performed before. The main difference here is the use of 2 sensor units rather than just one. The analysis is done initially for the top sensor and then for the bottom one. The sensor data were again obtained from the sensors and were added to an Excel file for ease of analysis. Then, the plot of Data no. vs magnetic field strength readings were plotted to identify the point at which the EMP signal was registered. Figure 5.18 displays this plot (for the top sensor unit) and the spike at approximately data no 2300. The exact point at which the spike occurred was data no. 2294

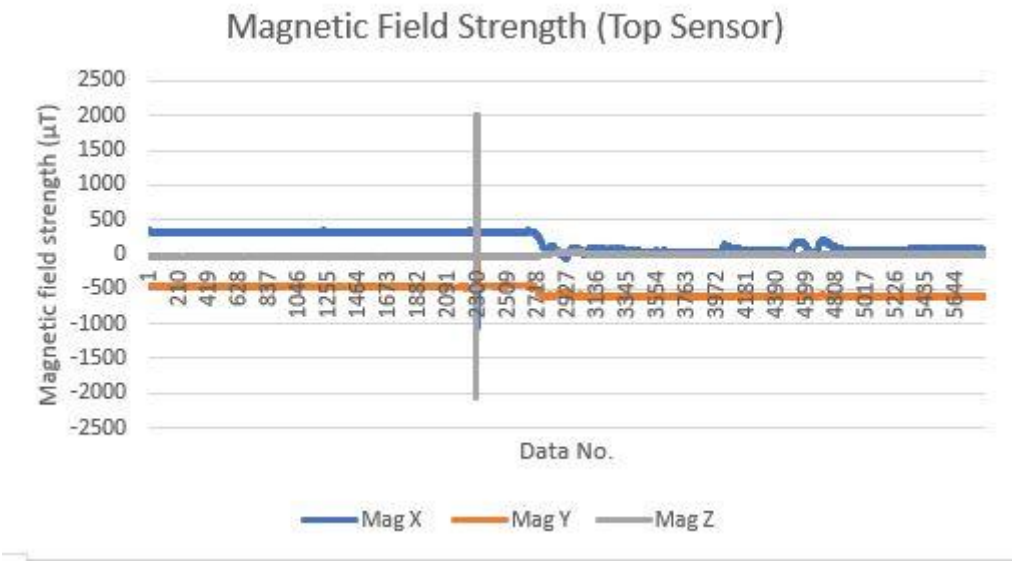

Figure 5. 18 Magnetic field strength plot of the top sensor

Then, the timestamp of the second image captured was added as the timestamp of Data no 2294. Then the consequent timestamps were obtained by adding 10 milliseconds to the timestamp of the previous data. Next, the plot of data no vs acceleration values were obtained to approximately figure out the start and end of the gait cycle. Figure 5.19 shows this plot. Unlike the plot in the drop test, this is slightly complicated due to movement of the sensors in all three axis.

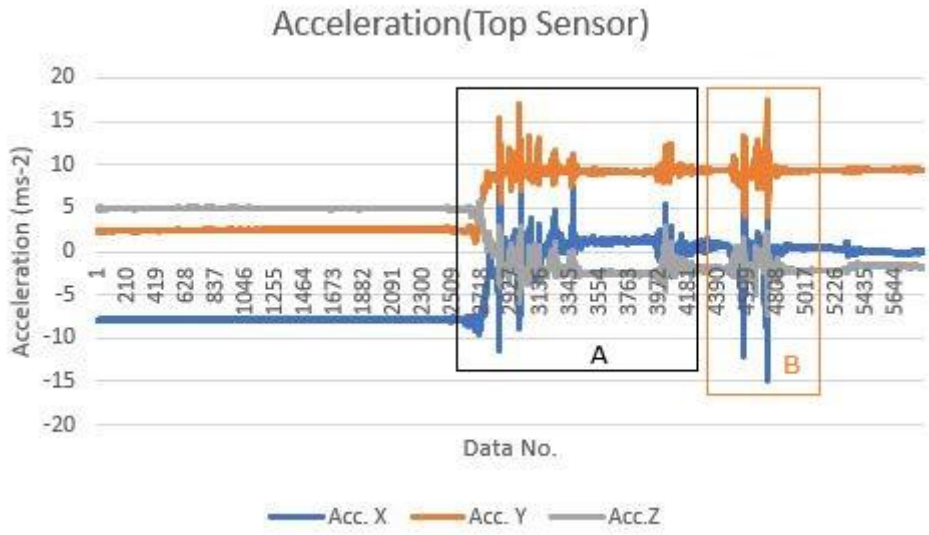

Figure 5. 19 Acceleration plot of the top sensor

The data from data no 2300 to 4181(shown by region A in figure 5.19) can be ignored as these correspond to the movement of the subject after EMP synchronization (position

shown in figure 5.17) to the start of the gait cycle position. The subject stood at rest for a few seconds when he reached the starting position. Then, he started walking (shown by region B in figure 5.19). The same method was used for the consequent tests as well. Next, the image frames obtained were analysed and the image frames corresponding to the beginning and the finish of the gait cycle where obtained. The gait cycle started at image frame 'ex613' and finished at image frame 'ex 705'. These image frames along with some intermediate image frames are shown in the following figures to illustrate a complete gait cycle.

Figure 5. 20 'ex613' on the left and 'ex705' on the right

Figure 5. 21 Some intermediate image frames

Next, the timestamps of the start and finish image frames were compared to the timestamps of the acceleration in x axis. In this case the x axis is parallel to the ground and is in the opposite direction to the direction of movement. Therefore, the acceleration values start decreasing from the value at rest. Similar to the drop test analysis, the start data and end data have been highlighted in blue and orange colours respectively. The acceleration starts decreasing (goes negative) at data no 4480 with a timestamp of 11:34:30.701. At data no 4818 the acceleration values start stabilizing back to 'rest' values. The timestamp of this data is 11:34:34.081. The timestamps of the image

frames ‘ex613’ and ‘ex705’ have been compared to the top sensor unit data timestamps in the table 5.2. Figure 5.22 shows the image timestamps. Rows 625 to 700 have been hidden in the figure to show both timestamps in the same image.

	A	B	C
1	Image No	Image name	Time before capture (Timestamp 1)
614	612	ex612	11:34:30.696
615	613	ex613	11:34:30.724
616	614	ex614	11:34:30.804
617	615	ex615	11:34:30.820
618	616	ex616	11:34:30.844
619	617	ex617	11:34:30.852
620	618	ex618	11:34:30.890
621	619	ex619	11:34:30.920
622	620	ex620	11:34:30.950
623	621	ex621	11:34:31.010
624	622	ex622	11:34:31.030
701	699	ex699	11:34:33.886
702	700	ex700	11:34:33.926
703	701	ex701	11:34:33.950
704	702	ex702	11:34:34.028
705	703	ex703	11:34:34.068
706	704	ex704	11:34:34.088
707	705	ex705	11:34:34.108
708	706	ex706	11:34:34.128

Figure 5. 22 Timestamps of image frames ‘ex613’ and ‘ex705’

Table 5. 2 Top sensor and image frame timestamps compared

Position	**Timestamp of sensor data**	**Timestamp of image**	**Difference**
Start	11:34:30.701	11:34:30.724	23 milliseconds
Finish	11:34:34.081	11:34:34.108	27 milliseconds

The final task was to do the same for the bottom sensor. Since both the top and bottom sensor image frames captured were the same, the image timestamps for start and finish of the gait cycle will be the same. The bottom sensor unit, unlike the top sensor unit, was programmed to turn off after 2 minutes after turning on. This created additional 1 minute of data which was not useful and created issues while plotting graphs. Therefore, all data after data number 5850 (the amount recorded by the top sensor) were deleted for analysis purposes.

The magnetic field strength plot was created to find the approximate location of the electromagnetic pulse spike. The figure 5.23 shows the spike at approximately between data no. 1954 and 2171. Upon closer analysis the exact point is located as data no 2067. Again, the timestamp of the second image captured, 'ex1', was added as the timestamp of this data. The consequent timestamps were also obtained by adding 10 milliseconds to the previous timestamp.

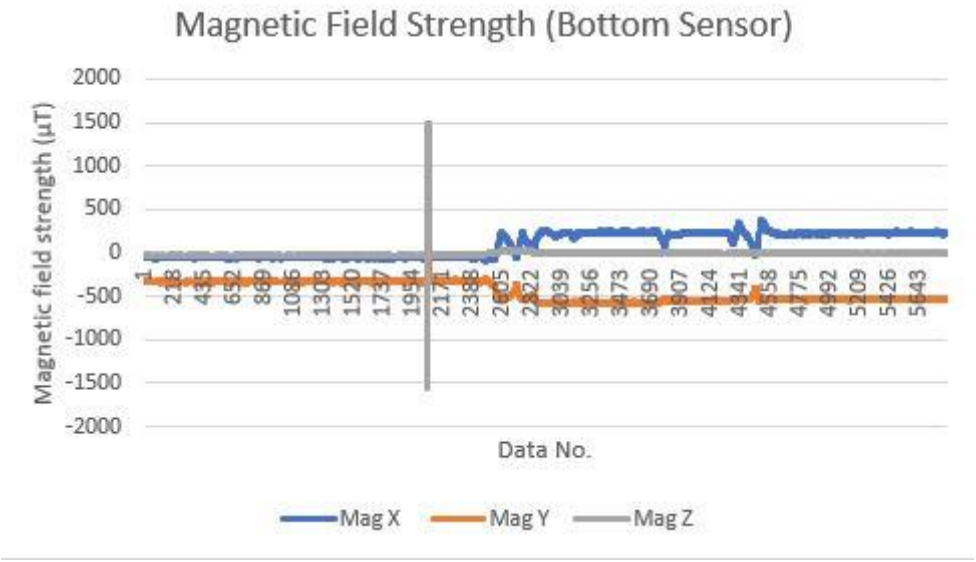

Figure 5. 23 Magnetic field strength spike of bottom sensor

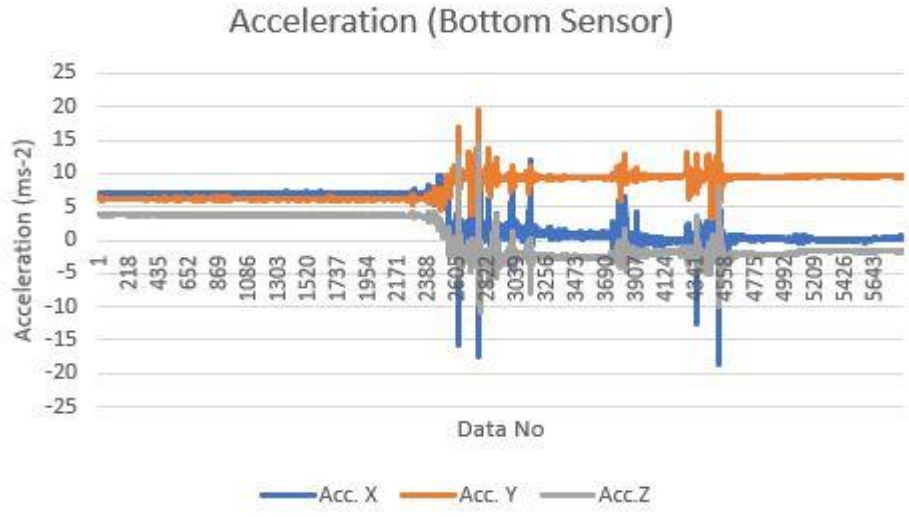

Figure 5. 24 Acceleration plot of the bottom sensor

The data no vs acceleration values plot is shown in the above figure 5. 24. Acceleration of x axis starts varying at data no. 4255 and the timestamp here is 11:34:30.721. The

gait cycle ends at data no. 4594 and the timestamp here is 11:34:34.111. The image timestamps have been compared to the bottom sensor timestamps in the table 5.3

Table 5. 3 Bottom sensor and image frame timestamps compared

Position	Timestamp of sensor data	Timestamp of image	Difference
Start	11:34:30.721	11:34:30.724	3 milliseconds
Finish	11:34:34.111	11:34:34.108	3 milliseconds

The accelerometer data (x axis) of both the top and the bottom sensor after the synchronization signal were also plotted to show that the EMP signal indeed did synchronize the data. The following figure 5.25 clearly shows (both signals having similar pattern) that both the sensor data have been synchronized.

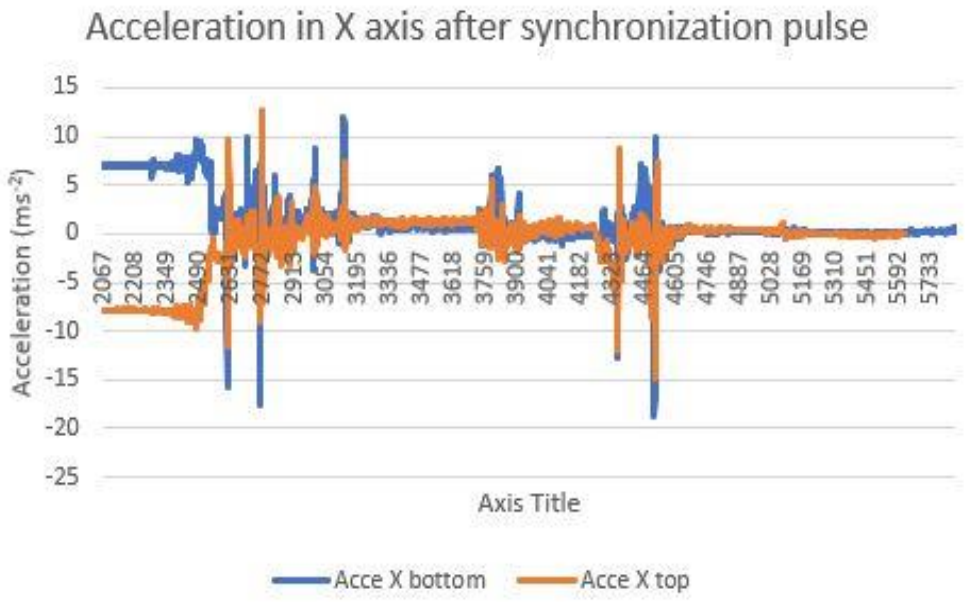

Figure 5. 25 Acceleration data of both sensor units after synchronization

The same type of gait cycle was carried out 9 more times in a similar way and the results have been tabulated in the table below. Test no. 8 had some wiring issues (loose wiring) with the EMP coil and didn't produce the electromagnetic pulse. Therefore, this test results were excluded from table 5.4.

Table 5. 4 Comparison of timestamps from further tests

Gait Test No.	Sensor Position	Position	Timestamp of sensor data	Timestamp of image	Difference
2	Top sensor	Start (ex390)	11:38:49.853	11:38:49.890	36 ms
		Finish (ex492)	11:38:52.993	11:38:52.949	43 ms
	Bottom Sensor	Start (ex390)	11:38:49.843	11:38:49.890	46 ms
		Finish (ex492)	11:38:52.973	11:38:52.949	24 ms
3	Top sensor	Start (ex471)	11:43:55.542	11:43:55.533	9 ms
		Finish (ex564)	11:43:58.782	11:43:58.793	1 ms
	Bottom Sensor	Start (ex471)	11:43:55.622	11:43:55.533	89 ms
		Finish (ex564)	11:43:58.792	11:43:58.793	1 ms
4	Top sensor	Start (ex306)	11:48:19.618	11:48:19.637	19 ms
		Finish(ex411)	11:48:23.208	11:48:23.127	81 ms
	Bottom Sensor	Start (ex471)	11:48:19.688	11:48:19.637	51 ms
		Finish (ex564)	11:48:23.198	11:48:23.127	71 ms
5	Top sensor	Start (ex497)	11:55:54.673	11:55:54.753	80 ms
		Finish (ex587)	11:55:58.193	11:55:58.213	20 ms
	Bottom Sensor	Start (ex497)	11:55:54.713	11:55:54.753	40 ms
		Finish (ex587)	11:55:58.263	11:55:58.213	50 ms
6	Top sensor	Start(ex623)	12:16:06.151	12:16:06.086	75 ms
		Finish(ex721)	12:16:09.271	12:16:09.352	83 ms

Table 5.4 Comparison of timestamps from further tests …… continued

Gait Test No.	Sensor Position	Position	Timestamp of sensor data	Timestamp of image	Difference
6	Bottom Sensor	Start(ex623)	12:16:06.041	12:16:06.086	45 ms
		Finish(ex721)	12:16:09.431	12:16:09.352	79 ms
7	Top sensor	Start (ex638)	12:19:55.780	12:19:55.806	26 ms
		Finish (ex734)	12:19:59.040	12:19:58.985	55 ms
	Bottom Sensor	Start (ex638)	12:19:55.750	12:19:55.806	56 ms
		Finish (ex734)	12:19:58.970	12:19:58.985	15 ms
9	Top sensor	Start (ex569)	12:30:38.135	12:30:38.189	54 ms
		Finish (ex671)	12:30:41.605	12:30:41.589	16 ms
	Bottom Sensor	Start (ex569)	12:30:38.045	12:30:38.189	144 ms
		Finish (ex671)	12:30:41.485	12:30:41.589	104 ms
10	Top sensor	Start (ex561)	12:36:39.105	12:36:39.089	16 ms
		Finish (ex652)	12:36:42.105	12:36:42.112	7 ms
	Bottom Sensor	Start (ex561)	12:36:38.995	12:36:39.089	91 ms
		Finish (ex652)	12:36:42.025	12:36:42.112	87 ms

Tests 1 to 5 were conducted with the sensors attached to the left leg of the subject. The subject walked from the right side of the image frame to the left. Tests 6-10 were conducted with the sensors attached to the right leg and walked from the left side of the image frame to the right. The test results show that the difference between the image frame timestamp and the sensor data timestamps are within 100ms. The only exception occurs in test number 9 bottom sensor where the differences are 144ms and 104ms.

These results show that the synchronization of camera image frames and IMU data within 100ms (except in one case). The method however is not 100 percent error proof. Since the IMU data that were analysed were the acceleration data, it can be quite difficult to pinpoint the exact instance at which the leg started moving or the point at which it is at rest.

SUMMARY

The first and main objective of this thesis was to synchronize image frames obtained from a camera, transmitted to the computer in real time, with data from IMU sensors which are not transmitted to the computer in real time. This objective was completed as shown by the results in the previous chapter. All the tests (box drop test and gait tests) have shown that the synchronization of the image frames with IMU data were within 100ms and only above 100ms in one case (test no 9) . As discussed previously, these differences occurred mainly due to the gap between the image acquisition rate and the sensor data acquisition rate. Also, the acceleration values used to find the start and end of the gait cycle could cause some errors. This is because an acceleration value of zero could mean 2 possible states. One is that the sensor is at rest. The other is that the sensor is moving at a constant velocity. The IMU sensors used in this thesis register acceleration values up to 4 decimal places and can capture even tiny movements. Due to these reasons, it is difficult to pinpoint the exact point at which the movement started.

There are a few further improvements that can be done in the future to make the synchronization more accurate. The use of a highspeed camera which can capture images at the same rate as the IMU sensor can drastically reduce the difference in timestamps between the images and the sensor data. This would also be very useful while analysing quick or minor movements of the sensors as the video can be slowed down to a certain extent without missing intermediate frames.

The second objective was to synchronize the image frames with the computer clock. The webcam used in this thesis did not contain an internal clock as many highspeed cameras such as the uEye camera do. Therefore, the image frames from the web camera had to be timestamped with the computer clock as the reference as it was the only clock available. However, the image capture delay test shown in the Tests and Results chapter can be used for a camera with its own clock to find out total delay (exposure delay + transmission delay) and thereby synchronize the image timestamps with the computer clock as the reference. This method would yield very accurate results for a camera which can capture images at a fixed frame rate.

This method consumed a lot of time during analysis because it was done manually by going through each and every image obtained in the test. But by conducting many tests and obtaining thousands of image frames, a computer vision/ machine learning system could be used in the future to figure out the delays automatically. This would make the system more efficient and would require specialized knowledge.

The third objective was to use a Python script to do camera setup, image collection, timestamping and synchronization. Everything except the synchronization part was successfully completed using the Python script. Synchronization part couldn't be completed here due to complexities in data analysis and had to be done manually. This part could be accomplished by a Python script in the future by employing several libraries which deal with data analysis. Also, a Python based GUI could be used to display, slow down, speed up, zoom in and out the image stream (video) and the corresponding data simultaneously. In the case of Gait analysis, this would make viewing and analysis of the data by as doctors or regular patients easier and convenient.

The final objective was to make the Python script simple and versatile enough so that cameras could be changed without a major overhaul in Python script. This objective too was accomplished successfully. The script used for image acquisition here can be used with almost any web camera. The script also could be used for highspeed and other external cameras but the camera setup, image acquisition and timestamping procedure will have to be modified. New libraries that work with the camera will have to be imported and used based on the camera and its brand.

KOKKUVÕTE

Käesoleva töö esmane ja peamine eesmärk oli sünkroniseerida kaamerast saadud ja reaalajas arvutisse edastatavad kaadrid IMU (inertsiaalne mõõteseade) andurite andmetega, mida ei edastata arvutisse reaalajas. See eesmärk täideti, nagu näitavad eelmise peatüki tulemused. Kõik testid (kasti langemise test ja kõnnakutestid) näitasid, et kaamera kaadrite sünkroniseerimisel IMU andmetega oli nihe 100 ms piires ja ainult ühel juhul üle 100 ms (test nr 9). Nagu varem arutletud, ilmnesid need erinevused peamiselt piltide salvestamise sageduse ja anduri andmete salvestamise sageduse erinevuse tõttu. Samuti võivad kõnnaku alguse ja lõpu leidmiseks kasutatud kiirendusväärtused põhjustada mõningaid vigu. Seda seetõttu, et nullkiirenduse väärtus võib tähendada kahte võimalikku olekut: andur on puhkeasendis või andur liigub ühtlasel kiirusel. Selles töös kasutatud IMU-d registreerivad kiirenduse väärtused kuni 4 kümnendkoha täpsusega ja suudavad tuvastada isegi väikeseid liigutusi. Nendel põhjustel on raske täpselt kindlaks teha liikumise alguspunkti.

Sünkroonimise täpsemaks muutmiseks saaks tulevikus veel paar täiendust teha. Kiire kaamera kasutamine, mille kaadrisagedus ühtiks sensorite andmesalvestuse kiirusega, võib drastiliselt vähendada kaadrite ja andurite andmete ajatemplite erinevust. Samuti oleks see väga kasulik andurite kiirete või väiksemate liikumiste analüüsimisel, kuna videot saab aeglustada ilma kaadreid kaotamata.

Selle töö teine eesmärk oli sünkroonida video kaadrid arvuti kellaga. Selles lõputöös kasutatud veebikaamera ei sisaldanud sisemist kellamoodulit, mida kasutavad paljud suure kiirusega kaamerad, näiteks uEye kaamera. Veebikaamera kaadrid tuli ajatempliga varustada arvuti kella abiga, kuna see oli ainus saadaval olev kell süsteemis. Peatükis *Tests and Results* näidatud pildi jäädvustamise viivituskatset saab aga kasutada oma kellaga kaamera jaoks, et teada saada kogu viivitus (särituse viivitus + andmete edastamise viivitus) ja sünkroonida seeläbi kaadrite ajatemplid arvutikellaga kui võrdlusalusega. See meetod annaks väga täpseid tulemusi kaamera jaoks, mis kasutab püsivat kaadrisagedust.

Kasutatud meetodi puhul kulus analüüsiks palju aega, kuna see tehti käsitsi, vaadates läbi kõik testi käigus salvestatud kaadrid. Paljude testide läbiviimisel ja tuhandete piltide hankimisel võiks edaspidi kasutada arvutinägemist / masinõppet, et viivitused automaatselt välja selgitada. See muudaks süsteemi tõhusamaks ja nõuaks eriteadmisi. Kolmas eesmärk oli Pythoni skripti kasutamine kaamera seadistamiseks, piltide kogumiseks, ajatemplite lisamiseks ja sünkroonimiseks. Kõik, välja arvatud

sünkroniseerimine, õnnestus Pythoni skripti abil edukalt lõpule viia. Andmeanalüüsi keerukuse tõttu ei saanud siin sünkroniseerimise osa lõpule viia ja see tuli teha käsitsi. Selle osa saab tulevikus Pythoni skripti abil täita, kasutades mitut andmeanalüüsiga tegelevat teeki. Samuti võiks Pythoni põhist GUI-d kasutada pildivoo (video) ning vastavate andurist saadud andmete kuvamiseks, aeglustamiseks, kiirendamiseks, suurendamiseks ja vähendamiseks jne. Kõnnaku analüüsi puhul muudaks see andmete vaatamise ja analüüsi arstide või tavapatsientide poolt lihtsamaks ja mugavamaks.

Lõppeesmärk oli muuta Pythoni skript piisavalt lihtsaks, aga samas mitmekülgseks, et saaks kasutada erinevaid kaameraid, ilma vajaduseta Pythoni skripti tõsiselt muuta. Ka see eesmärk saavutati edukalt. Käesolevas töös piltide saamiseks kasutatud skripti saab kasutada peaaegu iga veebikaameraga. Skripti võiks kasutada ka kiirete ja muude väliste kaamerate jaoks, kuid sellisel juhul tuleb kaamera seadistust, piltide hankimise ja ajatempli protseduuri muuta, vastavalt kasutatavale kaamerale. Samuti tuleb importida vastava kaamera jaoks mõeldus teegid.

LIST OF REFERENCES

[1] H. Y. Kim, "Modeling and tracking time-varying clock drifts in wireless networks," no. August, 2014.

[2] "Electromagnetic-radiation absorption by water." https://www.researchgate.net/publication/321740338_Electromagnetic-radiation_absorption_by_water (accessed Oct. 26, 2020).

[3] U. M. Qureshi *et al.*, "RF path and absorption loss estimation for underwaterwireless sensor networks in differentwater environments," *Sensors (Switzerland)*, vol. 16, no. 6, p. 890, Jun. 2016, doi: 10.3390/s16060890.

[4] J. M. Hovem, "Underwater acoustics: Propagation, devices and systems," *J. Electroceramics*, vol. 19, no. 4, pp. 339–347, 2007, doi: 10.1007/s10832-007-9059-9.

[5] Q. Wang, H. N. Dai, Q. Wang, M. K. Shukla, W. Zhang, and C. G. Soares, "On connectivity of UAV-assisted data acquisition for underwater internet of things," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5371–5385, 2020, doi: 10.1109/JIOT.2020.2979691.

[6] M. WHITTLE, "Applications of gait analysis," in *Gait Analysis*, Elsevier, 1985, pp. 177–193.

[7] S. A. Bridenbaugh and R. W. Kressig, "Laboratory review: The role of gait analysis in seniors' mobility and fall prevention," *Gerontology*, vol. 57, no. 3, pp. 256–264, 2011, doi: 10.1159/000322194.

[8] "Exposure time - SmartRay." https://www.smartray.com/glossary/exposure-time/ (accessed Nov. 17, 2020).

[9] K. Skiadopoulos *et al.*, "Synchronization of data measurements in wireless sensor networks for IoT applications," *Ad Hoc Networks*, vol. 89, pp. 47–57, 2019, doi: 10.1016/j.adhoc.2019.03.002.

[10] J. Wahslen, I. Orhan, T. Lindh, and M. Eriksson, "A novel approach to multi-sensor data synchronisation using mobile phones," *Int. J. Auton. Adapt. Commun. Syst.*, vol. 6, no. 3, pp. 289–303, 2013, doi: 10.1504/IJAACS.2013.054830.

[11] E. Cippitelli *et al.*, "Time synchronization and data fusion for RGB-Depth cameras and inertial sensors in AAL applications," *2015 IEEE Int. Conf. Commun. Work. ICCW 2015*, pp. 265–270, 2015, doi: 10.1109/ICCW.2015.7247189.

[12] "Cristian's Algorithm - GeeksforGeeks." https://www.geeksforgeeks.org/cristians-algorithm/ (accessed Jan. 08, 2021).

[13] J. Nikolic *et al.*, "A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 431–437, 2014, doi: 10.1109/ICRA.2014.6906892.

[14] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor Self-calibration," *Int. J. Rob. Res.*, vol. 30, no. 1, pp. 56–79, 2011, doi: 10.1177/0278364910382802.

[15] M. Li and A. I. Mourikis, "Online temporal calibration for camera-IMU systems: Theory and algorithms," *Int. J. Rob. Res.*, vol. 33, no. 7, pp. 947–964, 2014, doi: 10.1177/0278364913515286.

[16] "Introduction to Kalman Filter and Its Applications | IntechOpen." https://www.intechopen.com/books/introduction-and-implementations-of-the-kalman-filter/introduction-to-kalman-filter-and-its-applications (accessed Oct. 26, 2020).

[17] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 1280–1286, 2013, doi: 10.1109/IROS.2013.6696514.

[18] "ARM® Cortex®-M0+ SAM D21G." https://ww1.microchip.com/downloads/en/DeviceDoc/SAM_D21_DA1_Family_DataSheet_DS40001882F.pdf (accessed May 14, 2021).

[19] "Bosch BMX160." https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmx160-ds0001.pdf (accessed May 14, 2021).

[20] "MS5837-02BA." https://www.te.com/commerce/DocumentDelivery/DDEController?Action=srchrtrv&DocNm=MS5837-02BA01&DocType=DS&DocLang=English (accessed May 14, 2021).

[21] "RV-3028-C7." https://www.microcrystal.com/fileadmin/Media/Products/RTC/Datasheet/RV-3028-C7.pdf (accessed May 14, 2021).

[22] "Industrial microSD Cards | Western Digital." https://www.westerndigital.com/products/commercial-removable-storage/industrial-microsd (accessed May 14, 2021).

[23] "50mAh Lithium polymer battery." https://www.tme.eu/Document/54b17ccb776f7893ef046d26519e6622/LP301020CL.40.pdf (accessed May 14, 2021).

[24] "IDS Software Suite - IDS Imaging Development Systems GmbH." https://en.ids-imaging.com/ids-software-suite.html (accessed May 14, 2021).

[25] "All Python and Arduino codes used in 'Synchronization of Camera and Inertial Measurement Unit' by Balakrishnan Guruprasath (195425MAHM) are available here." https://github.com/BalakrishnanGuruprasath/cameraimusync (accessed May 14, 2021).

[26] C. Monoli, J. F. Fuentez-Perez, N. Cau, P. Capodaglio, M. Galli, and J. A. Tuhtan, "Land and Underwater Gait Analysis Using Wearable IMU," *IEEE Sens. J.*, vol. 21, no. 9, pp. 11192–11202, 2021, doi: 10.1109/JSEN.2021.3061623.