

**IZMIR KATIP CELEBI UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**

**INCUBATOR MONITORING SYSTEM FOR NEWBORN INFANTS USING
RASPBERRY PI**



M.Sc. THESIS

Mehmet Hakan SELEK

Department of Computer Engineering

Thesis Advisor: Assistant Professor Yalcin ISLER

SEPTEMBER 2018

**IZMIR KATIP CELEBI UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**

**INCUBATOR MONITORING SYSTEM FOR NEWBORN INFANTS USING
RASPBERRY PI**



M.Sc. THESIS

**Mehmet Hakan SELEK
(601514001)**

Department of Computer Engineering

Thesis Advisor: Assistant Professor Yalcin ISLER

SEPTEMBER 2018

İZMİR KATİP ÇELEBİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

RASPBERRY Pİ KULLANILARAK BEBEK KUVÖZÜ İZLEME SİSTEMİ
GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Mehmet Hakan SELEK
(601514001)

Bilgisayar Mühendisliği Ana Bilim Dalı

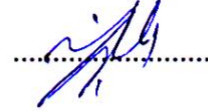
Tez Danışmanı: Dr. Öğr. Üyesi Yalçın İŞLER

EYLÜL 2018

Mehmet Hakan SELEK, a M.Sc. student of **IKCU Graduate School Of Natural And Applied Sciences**, successfully defended the thesis entitled “**INCUBATOR MONITORING SYSTEM FOR NEWBORN INFANTS USING RASPBERRY PI**”, which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :

Asst. Prof. Dr. Yalçın İŞLER
İzmir Katip Çelebi University



Jury Members :

Assoc. Prof. Dr. Yakup KUTLU
İskenderun Technical University



Asst. Prof. Dr. Mustafa YENİAD
Yıldırım Beyazıt University



Date of Submission : 07.08.2018
Date of Defense : 04.09.2018





To my family,



FOREWORD

First and foremost, I would like to express my special appreciation to my advisor Assistant Professor Yalçın İŞLER for guiding and supporting me over the period of my thesis study. I would also like to thank my committee members, Associate Professor Yakup KUTLU, Assistant Professor Mustafa YENİAD.

I would like to thank my family for the love, support, and encouragement against the difficulties I faced over the years.

Sept 2018

Mehmet Hakan SELEK



TABLE OF CONTENTS

	<u>Page</u>
FOREWORD	ix
TABLE OF CONTENTS	xi
ABBREVIATIONS	xiii
LIST OF TABLES	xv
LIST OF FIGURES	xvii
ABSTRACT	xix
ÖZET	xxi
1. INTRODUCTION	1
1.1 Causes of Premature Birth.....	1
1.2 The Best Environment to Grow up in Preterm Infants.....	2
1.2.1 The threats awaiting the preterm infants.....	2
1.2.1.1 Hypothermia.....	2
1.2.1.2 Apnea	4
1.2.1.3 Respiratory distress syndrome	5
1.2.1.4 Bronchopulmonary dysplasia.....	5
1.3 Incubator.....	6
1.3.1 History of the baby incubator.....	7
1.3.2 The structure of modern baby incubators.....	9
1.3.2.1 General parts of the incubator	10
1.3.3 How does the baby incubator work.....	12
1.3.4 Keeping track of baby incubator	12
1.4 Incubator Controller	13
2. DESIGN AND INSTRUMENTATION	15
2.1 Main Control Card.....	16
2.2 16 Bit Analog to Digital Converter	17
2.2.1 Continuous mode	18
2.2.2 Single-shot mode.....	19
2.3 DS18B20 Programmable Resolution 1-Wire Digital Thermometer	19
2.3.1 Working principle of DS18B20 temperature sensor.....	19
2.4 SHT 11	21
2.5 Preparing Raspberry Pi for Use.....	24
2.5.1 Compiling FFmpeg with hardware acceleration support.....	26
2.5.2 Compiling and installing Nginx with RTMP support.....	30
2.5.3 Apache server, PHP, MariaDB, phpMyAdmin and web page installation.....	37
2.6 Python Scripts.....	41
2.6.1 ADS1115 ADC python script	41
2.6.2 DS18B20 and DS18S20 temperature sensors	45
2.6.3 SHT11 temperature and humidity sensor.....	48

2.7 The Configuration and Content of the Web Page	49
2.8 Content and Configuration of the Web Page.....	51
2.8.1 Assigning trigger pins to the system	52
2.8.2 List of device-pin definition.....	53
2.8.3 Reports and graphs page	53
2.8.4 Add data unit page.....	55
2.8.5 Data unit list page.....	56
2.8.6 Add function page	56
2.8.7 Function list page	57
2.8.8 Add sensor data type page.....	58
2.8.9 Sensor data type list page	58
2.8.10 Undefined sensor list page	59
2.8.11 Add sensor page	60
2.8.12 Sensor list page.....	61
3. RESULT.....	62
3.1 Checking the System Usage	62
3.2 Checking the Functionality of the "Device Pin Options" Field.....	64
3.3 Controlling the Camera Live Stream.....	66
3.4 Checking the Functionality of the Sensor Area.....	67
4. CONCLUSION AND DISCUSSION.....	69
REFERENCES.....	71
.....	73

ABBREVIATIONS

3D	: Three Dimensional
ADC	: Analog-to-Digital Converter
ARM	: Advanced RISC Machine
BPD	: Bronchopulmonary Dysplasia
CPU	: Central Processing Unit
CSI	: Camera Serial Interface
CSV	: Comma-Separated Values
dB	: Decibel
DNA	: Deoxyribo Nucleic Acid
DSI	: Display Serial Interface
ECG	: Electrocardiogram
EXT4	: Fourth Extended Filesystem
FFmpeg	: Fast Forward MPEG
GB	: Gigabyte
GMIVH	: Subependymal Germinal Matrix with Intraventricular Hemorrhage
GND	: Ground
GPIO	: General-Purpose Input/Output
HDMI	: United States Dollar
HLS	: Http Live Streaming
HTTP	: Hyper Text Transfer Protocol
I2C	: Inter-Integrated Circuit
IC	: Integrated Circuit
ID	: Identity Document
IP	: Internet Protocol
LDR	: Light Dependent Resistor
LED	: Light-Emitting Diode
MHz	: Megahertz
MPEG	: Moving Picture Expert Group
NICU	: Neonatal Intensive Care Unit
OSX	: Operating System X
PGA	: Programmable Gain Amplifier
PHP	: Personal Home Page
RDS	: Respiratory Distress Syndrome
RH	: Relative Humidity
RPI	: Raspberry Pi
RTMP	: Real-Time Messaging Protocol
SBC	: Single Board Computer
SD	: Secure Digital
SMD	: Surface-Mounted Device
SPS	: Samples Per Second

SQL : Structured Query Language
SSH : Secure Shell
USB : Universal Serial Bus
USD : United States Dollar
VCC : VOLTAGE Collector to Collector
VDD : Voltage Drain Drain
WHO : World Health Organization



LIST OF TABLES

	<u>Page</u>
Table 2.1: Temperature/Data relationship.....	20
Table 2.2: Temperature conversion coefficients.	23
Table 2.3: Optimized V3 humidity conversion coefficients.	24
Table 2.4: Optimized V4 humidity conversion coefficients.	24





LIST OF FIGURES

	<u>Page</u>
Figure 1.1: von Ruehl warming tub.	8
Figure 1.2: Tarnier incubator.	9
Figure 2.1: Raspberry Pi 2 Single-Board Computer.	17
Figure 2.2: ADS1115 Analog to digital converter.	18
Figure 2.3: Temperature Register Format.	20
Figure 2.4: Pinout of DS18B20.	21
Figure 2.5: Sensor communication and reading flowchart.	23
Figure 2.6: apt-get update screenshot.	25
Figure 2.7: apt-get upgrade screenshot.	25
Figure 2.8: FFmpeg configuration.	27
Figure 2.9: FFmpeg compile screenshot.	28
Figure 2.10: The installation of the checkinstall package.	28
Figure 2.11: Entering a description for the deb package via checkinstall.	29
Figure 2.12: Sample deb package description.	29
Figure 2.13: Creating a checkinstall package and completion of the installation process.	30
Figure 2.14: Example of downloading Nginx, extracting it from compressed file and configuring it.	31
Figure 2.15: Sample output for Nginx configuration start.	31
Figure 2.16: Sample output for Nginx configuration end.	32
Figure 2.17: The start of the Nginx compilation process.	32
Figure 2.18: The end of Nginx compilation process.	32
Figure 2.19: Entering a description via checkinstall for the deb package for Nginx.	33
Figure 2.20: The description of the deb package for Nginx.	34
Figure 2.21: Completion of package creation and installation for Nginx with the checkinstall.	34
Figure 2.22: Nginx server configuration.	35
Figure 2.23: Configuration of Nginx for hls broadcast.	35
Figure 2.24: Nginx configuration for running rtmp server.	36
Figure 2.25: nginx.service file content.	36
Figure 2.26: Installation of Apache2.	38
Figure 2.27: Installation of MariaDB.	38
Figure 2.28: Installation of PhpMyAdmin.	39
Figure 2.29: User access permissions for folders.	39
Figure 2.30: Importing the SQL dump.	40
Figure 2.31: phpMyAdmin database screenshot.	40
Figure 2.32: NICU Monitoring System login screen.	41

Figure 2.33: Interfacing options:I2C.....	42
Figure 2.34: Enabling the I2C interface.....	43
Figure 2.35: The prompt after enabling the I2C interface.....	43
Figure 2.36: The accessible addresses that are on the I2C data line.....	44
Figure 2.37: ADS1115_READ.py file.....	44
Figure 2.38: Interfacing options 1-Wire.....	45
Figure 2.39: Enabling the 1-wire interface.....	45
Figure 2.40: The prompt for the confirmation of one-wire being enabled.....	46
Figure 2.41: One-wire bus devices.....	46
Figure 2.42: Sample output from reading sensors.....	46
Figure 2.43: DS1820.py file.....	47
Figure 2.44: SHT11.py file.....	48
Figure 2.45: SensorActivityLog table.....	49
Figure 2.46: CalculationFunctions table.....	50
Figure 2.47: SensorActivity table.....	50
Figure 2.48: Sensors table.....	51
Figure 2.49: Screenshot of the main page.....	51
Figure 2.50: Raspberry Pi GPIO pin chart.....	52
Figure 2.51: Device add pin page.....	53
Figure 2.52: Device pin list page.....	53
Figure 2.53: Reports and graphs page.....	54
Figure 2.54: Sensor activity list screenshot.....	54
Figure 2.55: Sensor activity graph screenshot.....	55
Figure 2.56: Add Data Unit Page screenshot.....	56
Figure 2.57: Data unit list page screenshot.....	56
Figure 2.58: Sample function entry to the add function page.....	57
Figure 2.59: Function List page screenshot.....	57
Figure 2.60: Add sensor data type page screenshot.....	58
Figure 2.61: Sensor data type list page screenshot.....	59
Figure 2.62: Undefined sensor list page screenshot.....	59
Figure 2.63: Undefined sensor add page screenshot.....	60
Figure 2.64: Add sensor page screenshot.....	61
Figure 2.65: Sensor list page screenshot.....	61
Figure 3.1: “htop” screenshot.....	62
Figure 3.2: “df -h” command screenshot.....	63
Figure 3.3: The state of the system while there is a MariaDB database query in progress.....	63
Figure 3.4: Database table structure.....	63
Figure 3.5: Pin triggering sample.....	64
Figure 3.6: Gpio pin states.....	65
Figure 3.7: “gpio -g read” command pin states.....	65
Figure 3.8: The circuit which is set-up on the breadboard.....	65
Figure 3.9: Live stream sample.....	66
Figure 3.10: The ts files that are being used while hls is in progress.....	67
Figure 3.11: Sensor area.....	67
Figure 3.12: The 2 nd state on AN0 probe.....	68

INCUBATOR MONITORING SYSTEM FOR NEWBORN INFANTS USING RASPBERRY PI

ABSTRACT

Preterm newborns are unlikely to live without any additional support. Babies born in this way need special care and a place that would imitate their mother's womb as much as possible. The environmental conditions like the temperature, humidity, noise, light intensity, etc. need to be carefully managed. In addition to managing the environment, it is necessary to record vital data such as baby's temperature and pulse and record any changes so that they can be examined later. Since the systems that carry out these mentioned controls are very expensive, there are still hospitals that do not have baby incubators around the world.

In this work, easily available and highly accurate components such as the DS18B20, SHT11, and ADS1115, are used. By using Raspberry Pi which is a single card computer that can be supplied easily, data can both be saved from sensors and monitored live. At the same time, the baby can be monitored live through the camera connected to the system. Also, the devices connected to the system, such as Heater and Humidifier can be triggered remotely.



RASPBERRY Pİ KULLANILARAK BEBEK KUVÖZÜ İZLEME SİSTEMİ GELİŞTİRİLMESİ

ÖZET

Preterm yenidoğanların, herhangi bir ek destek olmadan hayata tutunma olasılığı oldukça düşüktür. Bu şekilde doğan bebeklerin gelişimlerini tamamlayabilmeleri için özel bakıma, anne karnında geliştikleri ortama en yüksek oranda benzeyen bir yere ihtiyaçları vardır. Bulunacakları ortamın sıcaklığı, nem oranı, gürültü oranı, ışık oranı vb. çeşitli ortam koşullarının çok iyi ve hassas bir şekilde yönetilmesi gerekmektedir. Ortamın yönetilmesine ek olarak bebeğin ısı, nabızı gibi yaşamsal verilerin izlenmesi ve değişikliklerin sonradan incelenbilmesi için kayıt altına alınması gerekmektedir. Bahsedilen kontrolleri gerçekleştiren sistemler ise yüksek maliyetli oldukları için dünya üzerinde bebek kuvözüne sahip olmayan hastaneler bulunmaktadır.

Bu çalışma sonucunda, DS18B20, SHT11 gibi hassas ölçüm yapabilen sensörler ve ADS1115 ADC ile 4 kanallı 16 bit çözünürlüğünde analog okuma yapabilen yüksek hassasiyete sahip cihazlardan, kolayca temin edilebilecek tek kart bilgisayar Raspberry Pi kullanılarak, veri toplama, canlı olarak sensör verilerini izleme, sisteme bağlanan kamera aracılığıyla bebeğin canlı olarak izlenmesine ve kuvöze bağlanan, Isıtıcı, Nemlendirici gibi cihazların uzaktan tetiklenmesine olanak veren sistemin kolay bir şekilde inşa edilmesi sağlanmıştır.



1. INTRODUCTION

1.1 Causes of Premature Birth

Despite recent developments in the field of medicine, it is still unclear what causes of birth and how it begins [1]. If the factors that initiate normal birth are activated before their usual time, it will naturally cause the threat of premature birth. There are some factors that are thought to cause premature birth.

Infection comes first. The urinary tract, especially in the last stages of the pregnancy period, may start the birth with certain substances that infect them [1]. In addition, such infections may result in early rupture of the membrane by decreasing the resistance of the amniotic membrane. If this is the reason for birth, it is called "Preterm Premature Rupture of Membranes", which is an important factor in premature birth [2-3].

Another factor that causes premature birth is multiple pregnancies. Since the uterus grows too much, labor pains may start early [4]. In polyhydramnios cases, premature birth may take place [4].

Congenital uterine anomalies are another cause of late abortions and premature births [4]. However, there is no rule that any woman with this type of deformity will have a premature birth. These patients have only an increased risk.

Uterine myomas can initiate preterm labor by decreasing volume in the uterus [5].

The most preventable cause of premature birth is smoking. Preterm birth is one of the major hazard awaiting mothers who smoke [6].

1.2 The Best Environment to Grow up in Preterm Infants

1.2.1 The threats awaiting the preterm infants

1.2.1.1 Hypothermia

Neonatal hypothermia is a serious life-threatening condition characterized by the fact that the body temperature measured rectally in newborns is below 36 ° C. According to the definition of the World Health Organization (WHO), mild hypothermia is defined as a body temperature of 34C-36 ° C, a moderate hypothermia of 32-34 ° C or a hypersensitive hypothermia of 32 ° C [7]. If the babies are sick and they become hypothermic, this will increase their existing problems.

There is a close relationship between hypothermia and prematurity. The risk of hypothermia increases as the week of gestation becomes smaller. Premature babies have a lower body weight and therefore they have significantly less body heat [8]. Higher surface area and body weight is an important factor in increasing heat loss. In addition, subcutaneous fat tissue which is to be used for heat and isolation is less than that of mature infants. Muscle tissue that is used to generate heat via contraction is not sufficiently developed. Vasomotor responses are weaker than those of mature infants, so heat loss occurs faster.

Undeveloped epidermis in pregnancies less than 30 weeks, loss of heat due to evaporation and loss of trans-epidemic fluid have not been reported. Newborns in this condition are at high risk of fluid loss through the skin, and temperature instability, dehydration, electrolyte imbalance causes heat and calorie loss at the same time [9].

Environmental humidity reduces transepidermal water loss in these babies and helps to regulate temperature. At the same time, environmental humidity slows down the skin adaptation process to the baby's outer environment. In this case, the condition of the newborn should be kept in balance [10]. For neonates, heat and moisture are needed for epidemic DNA synthesis, which is beneficial as a stimulant to dry environment [11].

For pregnancies less than 30 weeks, NICU should be taken and those remaining below 2 weeks should be as moist as possible.

Babies in pregnancies less than 28 weeks:

They should be kept in an incubator with a humidity of 80% for 7 days.

At the end of the 7th day, humidity should be reduced by 5% per day.

The humidification should be stopped when the humidity level reaches 40%.

Babies in 28 - 29 + 6-week pregnancies:

They should be kept in an incubator with a humidity of 80%. After 1 day the humidity should be reduced by 5%. Humidification should be stopped when the humidity reaches 40%.

For babies born too early (up to 25 weeks), humidification process can be kept longer to avoid dehydration and hypothermic problems.

Undeveloped epidermis in pregnancies less than 30 weeks, loss of heat due to evaporation and loss of trans-epidermal fluid have not been reported. Newborns in this condition are at high risk of fluid loss through the skin, and temperature instability, dehydration, electrolyte imbalance causes heat and calorie loss at the same time.

Environmental humidity reduces transepidermal water loss in these babies and helps to regulate temperature. At the same time, environmental humidity slows down the skin adaptation process to the baby's outer environment. In this case, the condition of the newborn should be kept in balance. For neonates, heat and moisture are needed for epidermal DNA synthesis, which is beneficial as a stimulant to dry environment.

For pregnancies less than 30 weeks, NICU should be taken and those remaining below 2 weeks should be as moist as possible.

Babies in pregnancies less than 28 weeks:

They should be kept in an incubator with a humidity of 80% for 7 days.

At the end of the 7th day, humidity should be reduced by 5% per day.

The humidification should be stopped when the humidity level reaches 40%.

Babies in 28 - 29 + 6-week pregnancies:

They should be kept in an incubator with a humidity of 80%. After 1 day the humidity should be reduced by 5%. Humidification should be stopped when the humidity reaches 40%.

For babies born too early (up to 25 weeks), humidification process can be kept longer to avoid dehydration and hypothermic problems.

1.2.1.2 Apnea

Premature apnea is a problem that can be seen nearly %100 of infants who weigh less than 1000 grams until the gestational week reaches 29. If the premature apnea is accompanied by bradycardia, desaturation, or if it is necessary to give a stimulus for the onset of respiration, the underlying pathology must be investigate. Conditions that cause apnea in prematurity [12]:

1. Hypoxia, causes of hypovolemia
2. Bacteria, sepsis
3. Necrotizing enterocolitis
4. Intraventricular hemorrhage, Periventricular leukomalacia, hydrocephalus
5. Hypoxic ischemic encephalopathy
6. Patent Ductus Arteriozus and Right-to-left shunt
7. Gastroesophageal reflux
8. Hypoglycemia, hypocalcemia
9. Hyponatremia, acidosis
10. Hypothermia, hyperthermia
11. Anemia
12. Local infections
13. Causes of airway obstruction
14. Convulsions

1.2.1.3 Respiratory distress syndrome

Respiratory distress syndrome (RDS), which is histopathologically referred to as hyaline membrane disease, is a serious lung problem, especially in premature infants due to lack of endogenous surfactant in the lung [13]. Surfactant reduces airborne surface tension on the lungs, thereby preventing alveolar collapses in the exhalation and increasing compliance in the lung. The frequency of RDS is inversely proportional to the gestational week. Other risk factors for RDS are the infant's being male, cesarean birth, having a twin, family history, and the presence of diabetes mellitus in the mother [13]. The use of exogenous surfactant, which is used from the 1990s onwards and the development of ventilator therapy, has increased the chances of survival for RDS patients.

In follow-up of RDS, fluid-electrolyte balance affects baby early (first 72 hours) due to inadequate glomerulotubular function; at the same time diuresis is affected. At the same time, increased pulmonary vascular resistance is associated with decreased lymphatic drainage, interstitial edema, and delayed diuresis. One of the important factors determining prognosis in RDS is GMIVH [14].

1.2.1.4 Bronchopulmonary dysplasia

It was first described by Northway in 1967 [15]. BPD is the damage of lungs due to oxygen toxicity of the implanted lung, barotrauma, voluntary overloading of fluid, infections, inflammation, PDA, vitamin A deficiency.

Bronchopulmonary dysplasia (BPD) remains the most common complication associated with preterm delivery. The frequency of BPD has increased due to the increasing rate of the infants who live despite their low birth weight. While classic BPD is strongly associated with mechanical damage and oxygen toxicity, in the new BPD definition; it is associated with immaturity, perinatal infection and inflammation, patent ductus arteriosus and the developmental disorder of the alveolar capillaries.

The incidence of BPD increases as birth weight decreases. The incidence rate is 75% in babies weighing 700-800 g and 13% in babies who are born 1250-1500 g. In

babies that have BPD, there are clinical signs of chronic obstructive pulmonary disease.

Treatment suggestions include fluid restriction, diuretic therapy, oxygen supplementation, increase in caloric value, bronchodilators and steroid treatments and the increase of hematocrit to a rate greater than 40%. Steroid therapy has been shown to increase ventilatory requirements, duration of extubation and oxygen demand, and lung compliance in infants who are at the risk of BPD [15].

1.3 Incubator

Premature babies need a sterile atmosphere at a constant temperature, a certain degree of nourishment, a fresh air purified by an antibacterial filtration, a silence that their sleep will not break apart, a doctor's supervision and nursing care since they cannot maintain their body temperature at the required level and stability. Baby incubators are produced to meet this requirement.

Incubators can be defined as a special box with sound insulation, which are usually transparent, sterile, having electronic hardware that are designed to support babies who are born prematurely or with problems the ability to continue their life function without assistance, allowing to monitor and maintain the temperature as the babies are accustomed to in the womb while also providing a clean environment with humidity and clean air with antibacterial filtration.

Care for babies with respiratory problems:

- Risky pregnancy (diabetes, high blood pressure, kidney failure)
- Care for babies such as blood clotting or pregnancy poisoning
- Treatment of babies who are not oxygenated at birth (asphyxia)
- Applying respiratory device support to the lung enhancement agent (surfactant) for the lungs of immature infants
- Treatment of infected infants
- Treatment of infants of elderly or young mothers
- Phototherapy of jaundiced babies, and diseases such as blood fungus

are treated in incubators.

The incubator basically consists of a cabinet part with a transparent cover, a controller unit and body sections that perform functions such as incubation heating, ventilation, and humidification. There should also be a scale that weighs when the baby is lying in the cabin section, an x-ray filament that can be inserted and removed without disturbing the baby, and mechanical mechanisms for moving or turning the baby up or down.

1.3.1 History of the baby incubator

Ancient knowledge suggests that keeping the new-born cold rather than hot increases the possibility of death. Before the 19th century, preterm babies were wrapped in swaddling clothes in several layers and kept in a warm environment such as in her mother's cuddle or near a stove [16].

In 1835, Johann Georg von Ruehl developed the first heating tube to protect the heat of newborns (Figure 1.1). This tube had a double walled iron structure and was open. The lukewarm water was able to heat the surface by filling the volume between the two walls. Variations of Ruehl's design were used as maintenance equipment before the year 1878 [16].

The history of the development of the incubator goes back to Egypt until centuries ago. Artificially refining and keeping the eggs was a secret and a special Egyptian method. Before 1799, these secrets were not introduced to Europe until Napoleon's expedition. Professor Stephen Tarnier saw a chicken incubator at the Paris Zoological Garden in 1878. The incubator was in an ancient Egyptian design and was made by Odile Martin. The professor focused on how this device could be used with modifications to keep the preterm newborns warm [16]. Odile Martin made an incubator which was similar to the modern closed-incubator (Figure 1.2).

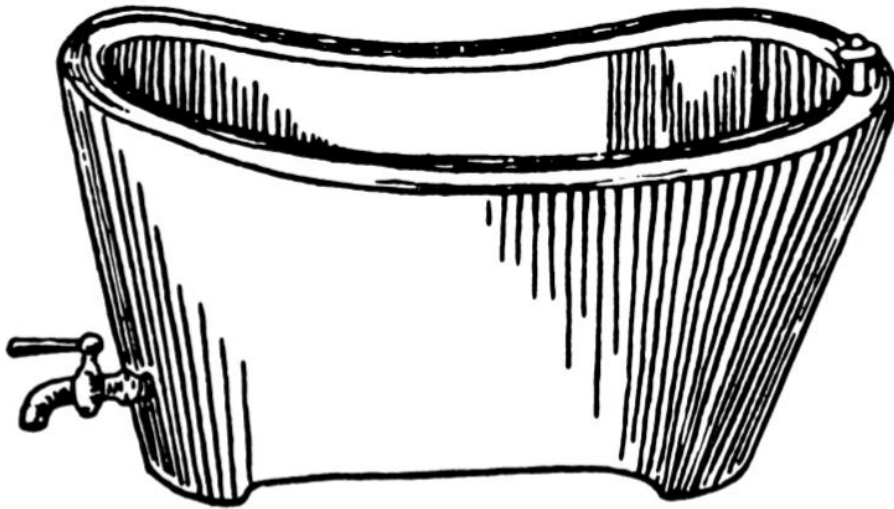


Figure 1.1: von Ruehl warming tub.

The Tarnier incubator was a very simple design. It had two compartments and was double walled. The walls were insulated to reduce heat loss. The newborns were put in the upper compartment. The upper lid of the upper compartment could be lifted and was operable. The heat of the upper compartment was provided by the water in the lower compartment. The water flow between the pipes and the water reservoir and the heater was provided through heating the water in a water, alcohol, or gas-powered heater. The air in the cold compartment went down and reached the heating compartment and the hot air was displaced by the cold air and rose up to the upper compartment [16].

Every knowledge that is gathered and development that has taken place throughout history has required manufacturers to develop and modify baby incubators in relation to the problems that they faced.

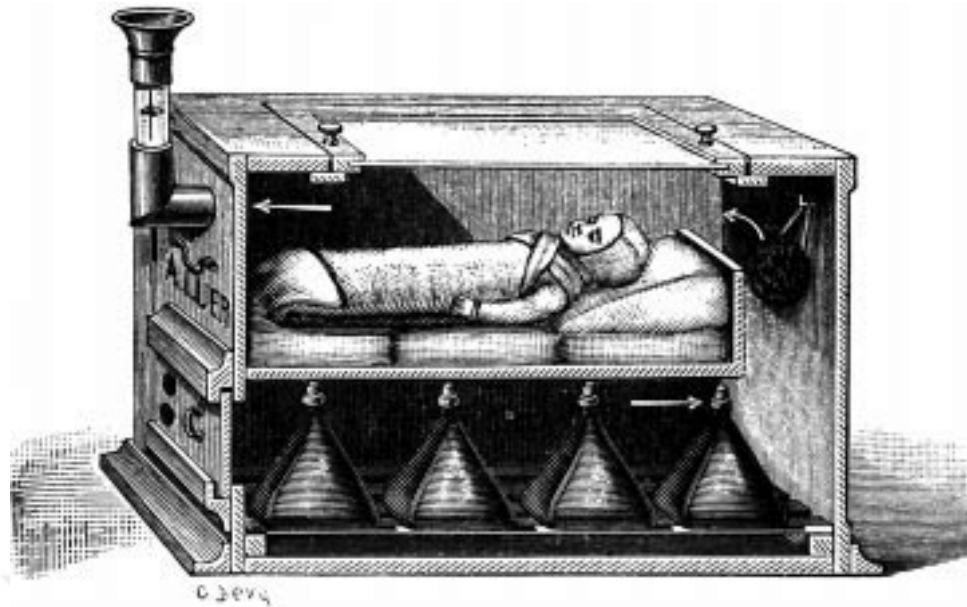


Figure 1.2: Tarnier incubator.

1.3.2 The structure of modern baby incubators

There are two types of incubators: stationary incubators and transportable incubators. And they are divided into protection, care, monitoring and treatment incubators.

Protection: In some cases, a sterile environment is required in order to protect the newborn infant from external effects. During the baby's development process, external assistance is kept on these devices for a certain period of time.

Care: These incubators provide support for the care and development of newborns.

Monitoring: In these incubators, vital functions such as body temperature, heart rate, respiratory status and pulse rate of newborns are continuously recorded with other medical devices integrated into the incubator and monitored and intervened if necessary.

Treatment: These incubators are equipped with an oxygen supplement to help with the breathing of newborns. In cases of physiological jaundice, a special treatment called phototherapy is used. The light used in this therapy can either be a fluorescent or a new generation LED.

There are also open baby beds called "heat beds" to keep or raise the body heat of newborns. Phototherapy can also be done in these beds.

Apart from the incubators described above, open infant beds are frequently used neonatal care devices, that are called neonatal intensive care beds, which are heated from the bottom or side with heating lamps, for the purpose of raising body temperature or keeping body heat stable and maintaining intensive care. These devices have integrated x-ray devices, scales, aspirator sets, vacuum sets and dampening units.

A standard transport incubation system consists of a transport tray, an O₂ cylinder, an air cylinder, a serum suspension, and an optional additional cylinder protection rail. The system has four shock absorbers for absorbing shocks and other vibrations and can be positioned at three different levels according to the position of the user. Depending on the suitability and technical analysis, other accessory models and additional parts can be connected to the incubator. Full observation, access, and security are provided with the help of a double-walled enclosure which reduces radiant heat loss, a pull-out bed for breathing, ECG cordless ventilation hoses for entry without occlusion, a front access panel and three input channels. On the microprocessor control panel, there are wide LED indicators for alarms, ambient temperature and baby skin temperature, sound level below 60 dBA, low and high air temperature, air circulation failure, energy failure, power indicator, battery charge, and heating level. With the automatically recharged life module, there is a continuous transportation guarantee of up to 4 hours. The AC 110/220 Volt output and DC12 Volt allow the battery to operate at both voltages without damaging the battery.

In ambulances, the transport incubation system is possible in two ways. There is a removable binding system for ambulances. This connection is also temporarily connected to the vehicle floor to ensure that the enclosure locks securely. In the stretcher connection, the incubator is temporarily connected to a standard stretcher. After the newborn baby has transported, the incubator can separate from the sled. The ambulance is equipped with accessories such as an infusion pump and a monitor.

1.3.2.1 General parts of the incubator

Control Panel: The easy-to-use control module is precisely controlled by an intelligent control module which controls the temperature level, humidity, and

oxygen in the patient compartment. All monitored information is updated graphically on the screen for 24 hours. The baby's weight can be updated graphically for 8 days. Calibration and maintenance can be monitored through the system.

Incubator Controller: The vital functions of the incubator are controlled by the incubator controller. The incubator controller will be discussed in more detail later in this article.

Alarms and Alerts: Because the baby is a living being, they need to work properly in the incubator, make no mistakes, or warn the user nurse in error situations. For example, disconnecting the electricity is an important condition. In this case, the temperature will decrease as the heater does not work. In the event of a power failure, the controller must be able to operate by supplying power from a battery while alarming in both an audible and illuminated way. Failure of sensors that provide temperature measurement should be immediately detectable and cause an alarm. For this purpose, the presence of a double sensor in the same temperature measuring probe increases the reliability. The user is also warned if the temperature cannot be controlled for any reason, i.e. high or low. Even in the case of extremely high temperatures, a thermostat that works independently of the controller must be able to disable the heater. The controller must be able to warn the humidifier if there is insufficient water in the water reservoir or if the relative humidity in the cabin falls below a certain limit for some other reason. For this purpose, a low humidity limit must be programmable by the user as a parameter. If the air flow falls below the normal level, an alarm should be given. This error would occur if the ventilation fan fails to operate for any reason or if the ventilation ducts are severely obstructed. Above all, the controller must be highly reliable, self-testable and able to warn the user in case of failure.

Oxygen Module: The control valve controls the oxygen level in the environment by means of the supply system and optionally operates via the system which controls the preset limits.

Moisturizer Module: As standard, the Styrofoam is integrated, allowing it to feed itself for 24 hours and to control the humidity between 30% and 95% RH.

1.3.3 How does the baby incubator work

It is known that providing proper thermal protection to newborn babies increases their pace of development and resistance to diseases [16]. Keeping the baby warm is very important at a critical level and it is quite difficult to wrap the nest in protective baby form. The incubators are the devices used to care for these high-risk newborns and are designed to adjust the environmental conditions according to the specific needs of each newborn. There are many types of incubators with structural differences. They all have a bed for the baby to have a heated microclimate, and a logical system allows it to be kept within the limits of the ambient temperature. In some incubators, the microclimate is produced in a fixed-wall chamber; these baby incubators are called closed incubators. If the air is propagated through a metallic bobbin by means of a fan, these incubators are called forced convection incubators. There are also open incubators; these incubators have no walls. So, there is nothing to limit the environmental convective, so these incubators are realized by directing the radiant heater onto the bearing area. These caves are generally called open radiant incubators, radiant warmer beds, or radiant heaters.

Each different incubator has specific advantages. Convectively heated incubators are a much easier way to adjust the humidity of the cuvette air conditioning that caretakers are in their newborns. With the open radiant heater, newborns suffer more body fluid loss [17]. But the more complex liquid management for the baby in the open incubator is easier and the baby is physically more accessible. In some cases, such features are also important for the healing of sick infants. The baby incubators used now combine the advantages of these two cubes. a radiant warmer bed and covers can be lifted. On this track, they are operated in radiant heater mode, giving quick access to the baby and, if the door is placed, in convectively heated mode, better control for sound and humidity control is provided [16].

1.3.4 Keeping track of baby incubator

It is also necessary to pay attention to external factors that affect the inner environment of the incubator as well as the incubator so that the baby can complete the development well. In the literature Van den Berg mentioned the properties of a "womb-like" environment. Other authors have supported this finding [16].

The hourly average of background and transient sounds should not exceed 50 dB and the transient sounds should be a maximum of 70 dB. The volume level should be kept below 45 dB [18-19].

The amount of light to observe the baby in the NICU should not be higher than 100 foot-candles [20]. The ambient light should be kept between 10 - 600 lux [21-20].

Provide cycled light from 32 weeks of gestation or when the preterm infant begins to differentiate between sleep and wake cycles, to help develop diurnal rhythms [20].

Appliance quiet times of 2-3 h during daytime and 12 h during the evening [20].

1.4 Incubator Controller

Tracking systems were needed to follow the external environment and NICU internal environmental factors. We have already described the use of an incubator controller to monitor and regulate NICU internal factors. The incubator controller is very important for the creation of a "womb-like" environment.

The incubator controller controls an electric heater cycle to regulate and maintain the temperature inside the cabin or the skin temperature of the baby. For this purpose, sensors for measuring the ambient (air) temperature and a sensor for measuring the baby's skin temperature are available. A fan provides fresh air from outside the cabin and air circulation inside the cabin. Air drawn by the fan is circulated through a container filled with water and humidified on this surface.

In addition to its basic functions, the incubator controller also has the task of collecting, recording and reporting data. Parameters such as mains voltage, cabin temperature, skin temperature, relative humidity, baby body weight for the last 72 hours should be stored in memory. In order to perform these functions, the incubator controller must have the ability to connect to the computer. It may even be more useful to connect the incubators in the same local unit to a computer in the form of a local network. It may also be very useful for the incubator controller to voice some pre-recorded messages when an alarm condition arises when the incubator door is opened or in other important situations.

Thanks to the ever-evolving technology, infant incubators are now able to be connected to the network via remote connection. It is possible to read the incubation data from a center without disturbing the baby and to follow it remotely and at the same time, remote intervention is possible.



2. DESIGN AND INSTRUMENTATION

The designed device is composed of various parts. These sections;

1. Controller unit

- Main control card (Raspberry Pi 2)
- Trigger switches
- Power supply board

2. 16 bit analog to digital converter(ADS1115)

3. Digital sensors

- SHT11
- DS18B20

4. Camera module

5. Raspbian configuration

- RTMP server
- WEB server
- SQL server
- FFmpeg
- BUS configuration

6. Python scripts

- ADS1115 ADC IC
- DS18B20 and DS18S20 temperature sensors
- SHT11 temperature and humidity sensor

7. WEB Page

- Live monitoring system
- Assign new sensors and trigger pins
- Assign new formulas for sensors
- Monitoring data history

Environmental temperatures which are sensed by DS18B20 sensors and read over I2C bus by a running python script generate an “HTTP GET” request to a specified local PHP page for each sensor. This PHP page can be called as a “web service”. In this PHP page, functions call SQL queries for the saved data for each sensor.

Saved temperature data will be processed by an index page for generating alarms and showing incubator status.

The user can control incubator remotely by using the web interface also web interface can trigger an alarm if a problem occurred in the incubator. Additionally, the user can watch Infant via the web interface.

2.1 Main Control Card

The Raspberry Pi is a small single-board computer (SBC) developed by the Raspberry Pi Foundation.

The developed system uses a Raspberry Pi 2 version and the Raspberry Pi 2 specification can be found below:

- A 900MHz quad-core ARM Cortex-A7 CPU
- 1GB RAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video

- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot
- VideoCore IV 3D graphics core

The features offered to us by the Raspberry Pi 2 single card computer are sufficient to make the project (Figure 2.1).



Figure 2.1: Raspberry Pi 2 Single-Board Computer.

The embedded Linux system can run the python code that has been written to collect data from sensors. At the same time, embedded Linux system supports SQL and PHP, so a web interface can be developed. 16Bit Analog to Digital converter and DS18B20 Sensor are controlled by I2C communication and SHT11 Temperature and Humidity sensor is controlled by 2 wire communication bus using a protocol similar to, but not quite the same as the I2C protocol.

2.2 16 Bit Analog to Digital Converter

Analog data is converted with ADS1115 analog to digital converter that is capable of measuring 16-bit high range resolution. The sensor is I2C compatible. ADS1115 also incorporate a programmable gain amplifier (PGA) and a digital comparator. The ADS111x perform conversions at data rates up to 860 samples per second (SPS)

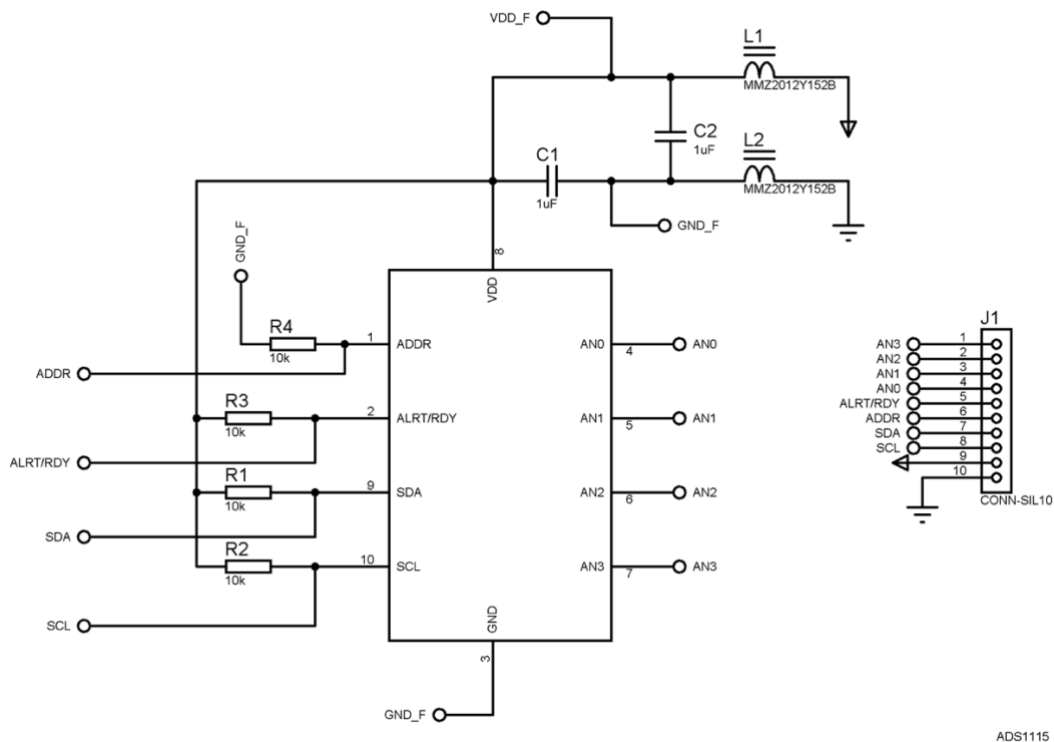


Figure 2.2: ADS1115 Analog to digital converter.

The diagram was made with Proteus. If the address is not connected to any output, by default it's value is set to 0. The reason that there is a ferrite bead on VCC and GND and these also being supported with a capacitor is to filter the static on the input of ADC circuit and to achieve a more healthy data reading(Figure 2.2).

There are 2 separate operating modes for ADS1115. The operating mode can be changed with the bit in the config register. These modes are as follows.

2.2.1 Continuous mode

For the ADS1115 to operate in continuous mode, the mode bit should be set to 0. When the circuit is set to this mode, it carries out a continuous conversation and reports the result to the conversation register. Whenever there is a new configuration setting is defined, the ongoing operation is carried out according to the old configuration setting and later starts according to the new setting.

2.2.2 Single-shot mode

For ADS1115 to work in single-shot mode, setting the config bit to 1 or resetting the device is enough. Also, when the device is connected to power, it starts working in single-shot mode by default.

2.3 DS18B20 Programmable Resolution 1-Wire Digital Thermometer

DS18B20 is a 1-Wire digital temperature sensor from Maxim IC. Reports degrees in Celsius with 9 to 12-bit precision, from -55 to 125 (+/-0.5). Each sensor has a unique 64-Bit Serial number etched into it - allows for a huge number of sensors to be used on one data bus.

The DS18B20 can be powered by between 3.0V and 5.5V so we can simply connect its GND pin to 0V and the VDD pin to +3.3V from the Raspberry Pi. This sensor has been included in many applications such as Thermostatic Controls, Industrial Systems, Consumer Products, Thermometers, Thermally Sensitive Systems.

2.3.1 Working principle of DS18B20 temperature sensor

The core functionality of the DS18B20 is its direct-to-digital temperature sensor. The resolution of the temperature sensor is user-configurable to 9, 10, 11, or 12 bits, corresponding to increments of 0.5°C, 0.25°C, 0.125°C, and 0.0625°C, respectively. The default resolution at power-up is 12-bit. The DS18B20 powers up in a low power idle state. To initiate a temperature measurement and A-to-D conversion, the master must issue a Convert T (44h) command. Following the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18B20 returns to its idle state. If the DS18B20 is powered by an external supply, the master can issue “read time slots” after the Convert T command and the DS18B20 will respond by transmitting 0 while the temperature conversion is in progress and 1 when the conversion is done.

The DS18B20 output temperature data is calibrated in degrees Celsius. The temperature data is stored as a 16-bit sign-extended two's complement number in the temperature register (Figure 2.3). The sign bits (S) indicate if the temperature is positive or negative: for positive numbers $S = 0$ and for negative numbers $S = 1$. If

the DS18B20 is configured for 12-bit resolution, all bits in the temperature register will contain valid data. For 11-bit resolution, bit 0 is undefined. For 10-bit resolution, bits 1 and 0 are undefined, and for 9-bit resolution bits 2, 1, and 0 are undefined. Table 2.1 gives examples of digital output data and the corresponding temperature reading for 12-bit resolution conversions.

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2 ³	2 ²	2 ¹	2 ⁰	2 ⁻¹	2 ⁻²	2 ⁻³	2 ⁻⁴
	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MS BYTE	S	S	S	S	S	2 ⁶	2 ⁵	2 ⁴

S = SIGN

Figure 2.3: Temperature Register Format.

The Temperature sensor has 3 pins (Figure 2.4) and it should be connected as follows:

- Vcc – 3.0 to 5V
- Gnd - Gnd
- Data-digital pin 2
- Pull up resistor of value 4.7kohm between data and supply voltage

Table 2.1: Temperature/Data relationship.

TEMPERATURE (°C)	DIGITAL OUTPUT (BINARY)	DIGITAL OUTPUT (HEX)
+125	0000 0111 1101 0000	07D0h
+85	0000 0101 0101 0000	0550h
+25.0625	0000 0001 1001 0001	0191h
+10.125	0000 0000 1010 0010	00A2h
+0.5	0000 0000 0000 1000	0008h
0	0000 0000 0000 0000	0000h
-0.5	1111 1111 1111 1000	FFF8h
-10.125	1111 1111 0101 1110	FF5Eh
-25.0625	1111 1110 0110 1111	FE6Fh
-55	1111 1100 1001 0000	FC90h

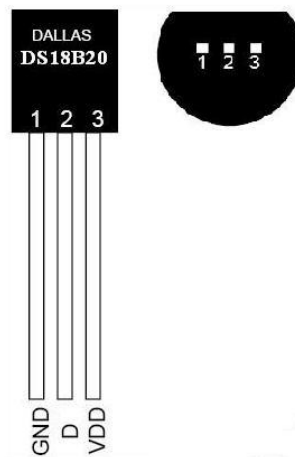


Figure 2.4: Pinout of DS18B20.

2.4 SHT 11

SHT11, which was produced by Sensirion Company of Switzerland with patented CMOSens technology. There are an internal 14 bit AD converter and serial communication units in the SHT11. It transmits temperature data on 14-bit resolution (14 bit is the default value, it is also possible to set it 12 bit) and humidity data on 12-bit resolution (12 bit is the default value, it is also possible to set it 8 bit) to the microcontroller. This SMD sensor is able to measure temperature from $-40\text{ }^{\circ}\text{C}$ to $+125\text{ }^{\circ}\text{C}$ with $\pm 0.5\text{ }^{\circ}\text{C}$ error margin. It also measures the humidity with $\pm \% 3.5$ error margin [22].

The humidity sensors are seamlessly coupled to a 14-bit-analog-to-digital converter and a serial interface circuit. This results in superior signal quality, a fast response time, and insensitivity to external disturbances [22].

The SHT11 integrates functions such as temperature sensing, humidity sensing, signal conversion, heater and A / D conversion into a single chip. The chip measurement stages are as follows:

- Humidity and temperature are converted into analog signals, by a capacitive polymer humidity sensor and a temperature sensitive element made of energy gap material.
- The electrical signal for amplification passes through the weak signal amplifier.

- The amplified analog signal passes through the 14 bit A/D converter for digital measurement.
- The digital signal is output through the two-wire serial digital interface.

The chip has a small size package, simple data interface, simple data protocol, and high accuracy. The serial interface of the SHT11 is optimized for power consumption and sensor reads. The sensor cannot be used over the I2C protocol. SHT11 two-wire serial protocol is different from the standard I2C protocol. It must work with SHT11 communication timing.

The sensor communication and the reading flowchart of SHT11 is shown in Figure 2.5. After initializing process done for the main program, communication reset process being applied. At this process, communication link renewed for avoiding timing faults between Raspberry Pi and SHT11. After communication reset process done, the command for starting measurement process is sent from the Raspberry Pi side. When the sensor gets a command from Raspberry Pi, then its starts measurement process. When the measurement process is done and data is ready for sent from SHT11, the Sensor data pin set its status low. At this point Raspberry Pi probes the data pin of SHT11. If data is ready to transfer, Raspberry Pi gets data from SHT11. If the data is not ready for the certain time, that means an error encountered during the process and it is impossible to get measurement data from SHT11. For avoiding this deadlock status of the process, the process falls timeout and starts from the beginning.

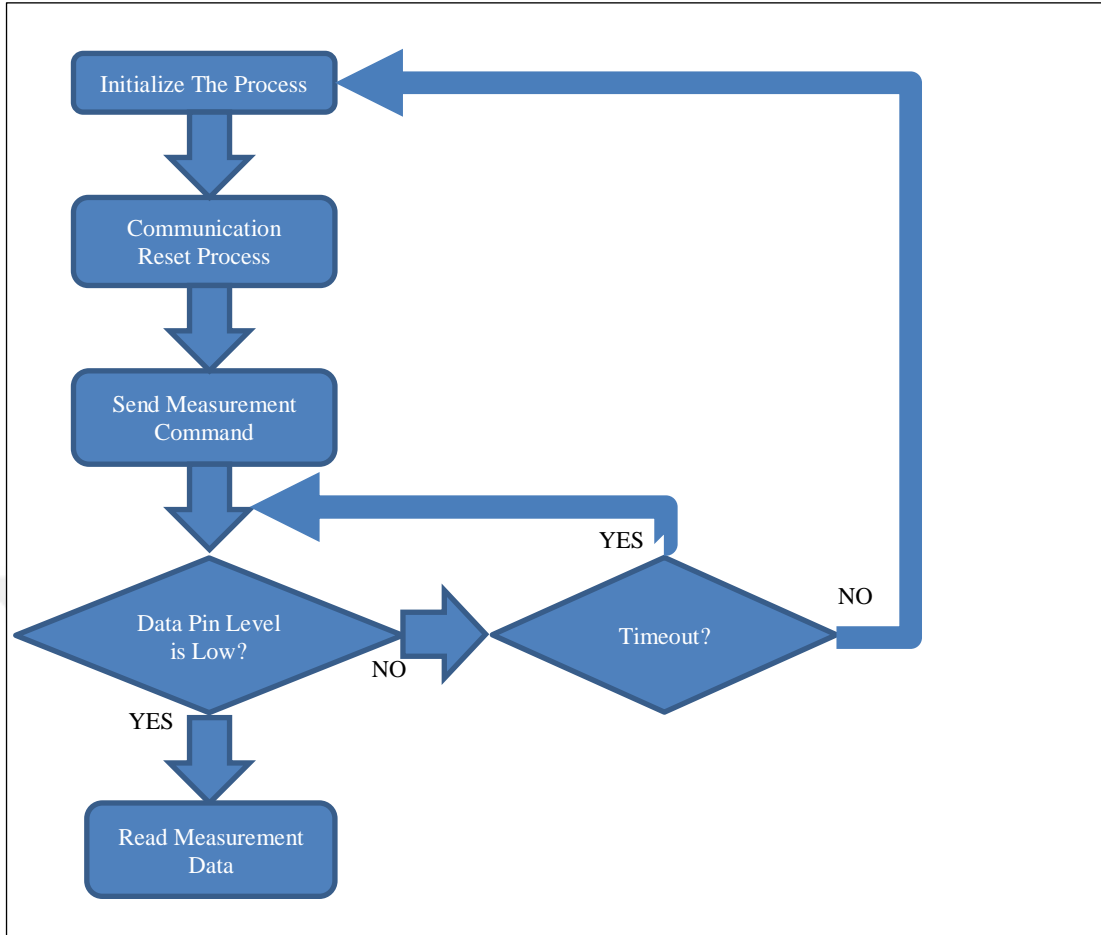


Figure 2.5: Sensor communication and reading flowchart.

After getting data from the SHT11 conversation is required to get meaningful values. SHT11 has good linearity owing to energy gap material. In this way temperature value can be calculated by using the following equation 2.1:

$$T = d_1 + d_2 * SO_T \quad (2.1)$$

d_1 and d_2 are constant which is shown in Table 2.2

Table 2.2: Temperature conversion coefficients.

VDD	d_1 (°C)	d_1 (°F)	SO_T	d_2 (°C)	d_2 (°F)
5V	-40.1	-40.2	14bit	0.01	0.018
4V	-39.8	-39.6	12bit	0.04	0.072
3.5V	-39.7	-39.5			
3V	-39.6	-39.3			
2.5V	-39.4	-38.9			

For humidity value calculation non-linear equation is required to be calculated as follows:

$$RH = C_1 + C_2 * SO_{RH} + C_3 * SO_{RH}^2 \quad (2.2)$$

SO_{RH} is measurement value of humidity which reads from SHT11. C_1 , C_2 , and C_3 are constants that vary due to the version of the sensor and resolution of the measurement data which are shown in Table 2.3 for V3 and Table 2.4 for V4.

Table 2.3: Optimized V3 humidity conversion coefficients.

SO_{RH}	C_1^*	C_2^*	C_3^*
12 bit	-4.0000	0.0405	-2.8000E-6
8 bit	-4.0000	0.6480	-7.2000E-4

Table 2.4: Optimized V4 humidity conversion coefficients.

SO_{RH}	C_1	C_2	C_3
12 bit	-2.0468	0.0367	-1.5955E-6
8 bit	-2.0468	0.5872	-4.0845E-4

2.5 Preparing Raspberry Pi for Use

In order for the Raspberry Pi to be usable, an image that contains one of the operating systems that is compiled specifically for the ARM needs to be loaded with an empty SD card. In this project, a Raspbian operating system image that is specially prepared for Raspberry Pi was loaded with a 16gb blank SD card and Raspberry Pi was booted over this image [23-24].

The Raspbian image used in the project is "2018-06-27-raspbian-stretch-lite" version. After the Raspbian image is loaded onto the SD card, a disk called "Boot" can be seen in the computer. This disk is the partition that allows the Raspberry to boot from the SD card. Because the second partition is in EXT4 format, it cannot be detected by the computer and is not visible.

After inserting the SD card into the Raspberry Pi, in order to access the Raspberry Pi via the IP, a file must be created inside the "Boot" directory that appears on the computer which is used to send the image to the Raspberry Pi and also this file

should not have any extensions and should be named "SSH". Since the computer used in the project is a computer hosting the OSX operating system, this file is opened by opening the terminal window of OSX and entering "touch /Volumes/Boot/SSH" command. The SD card is then inserted into the Raspberry Pi SD slot and then Raspberry Pi is connected to the power supply. After this point, Raspberry Pi's boot process begins.

When booting is complete, all services on Linux are now running and Raspberry Pi can now be accessed via SSH. After ensuring SSH access, first, "sudo apt-get update" and next, "sudo apt-get upgrade" commands should be entered via the terminal to make sure all packages are up to date.

With the "apt-get update" command entered on the terminal, the differences between the packages installed in the local system and the version in the package repository are searched and the list is updated (Figure 2.6). The "apt-get update" command does not perform any installation on the local system.

```
pi@raspberrypi:~ $ sudo apt-get update
Get:1 http://archive.raspberrypi.org/debian stretch InRelease [25.3 kB]
Get:2 http://raspbian.raspberrypi.org/raspbian stretch InRelease [15.0 kB]
Get:3 http://archive.raspberrypi.org/debian stretch/main armhf Packages [168 kB]
Get:4 http://archive.raspberrypi.org/debian stretch/ui armhf Packages [34.0 kB]
Get:5 http://raspbian.raspberrypi.org/raspbian stretch/main armhf Packages [11.7 MB]
Get:6 http://raspbian.raspberrypi.org/raspbian stretch/contrib armhf Packages [56.9 kB]
Fetched 12.0 MB in 21s (546 kB/s)
Reading package lists... Done
pi@raspberrypi:~ $
```

Figure 2.6: apt-get update screenshot.

After updating the package list with the "apt-get update" command, the "apt-get upgrade" command compares the version of the package list in the local system with the updated packet list (Figure 2.7). At the end of this process, if there is a lost package in the local system in use, it will be upgraded to the current package version.

```
pi@raspberrypi:~ $ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  ca-certificates dpkg dpkg-dev file libdpkg-perl libmagic-mgc libmagic1 libpam-systemd libraspberrypi-bin libraspberrypi-dev
  libudev1 patch raspberrypi-bootloader raspberrypi-kernel shared-mime-info systemd systemd-sysv tzdata udev
22 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 80.2 MB of archives.
After this operation, 239 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Figure 2.7: apt-get upgrade screenshot.

The command "apt-get install <package name or package names>" installs the current version of the package that is currently not installed but which is updated by the "apt-get update" command onto the local system.

2.5.1 Compiling FFmpeg with hardware acceleration support

FFmpeg is an open source free software that can convert any video format to another video format. FFmpeg supports almost all audio / video codecs (h264, h265, vp8, vp9, aac, opus, etc.), file formats (mp4, flv, mkv, ts, webm, mp3 etc.) and all streaming protocols , rtsp, hls, etc.) [25]. Since FFmpeg is a free open source project, the source code can be downloaded and compiled from ffmpeg.org or GitHub [26].

If FFmpeg is compiled and loaded as it is, it will not be able to use the GPU that Raspberry Pi 2 has on it, and if it is forced to stream in this way, the processor will be overloaded. In order to avoid this problem, such a heavy burden must be removed from the processor and handed to the GPU. In order for FFmpeg to be able to do this, it needs to be recompiled using the libraries of Raspberry Pi. The compilation process needs to be performed in the following steps, respectively.

First, in order to download FFmpeg source code from the GitHub repository, "git package" must be installed in the Linux system. In order to install "git package" on the system, "apt-get install git -y" command needs to be entered in the terminal. Once the "git package" has been installed on the system, the source code can now be downloaded from the git repositories and the code can be compiled with the configurations specified for the created system. Before the source code is downloaded to the system, it is recommended to create a folder, under the "/usr/" directory, called "build" to avoid any future problems.

In order to download FFmpeg source code to the "/usr/build" folder, first FFmpeg source code needs to be installed in the "build" folder by entering "cd /usr/build" command followed by "git clone <https://github.com/FFmpeg/FFmpeg.git>" Other packages and libraries are also required to compile after the process is complete. The installation of these packages and libraries is done with the command "sudo apt-get install build-essential libpcre3 libpcre3-dev libssl-dev libomxil-bellagio-dev -y".

Once the necessary libraries and packages have been installed, FFmpeg needs to be configured to use the Raspberry Pi's hardware. The required configuration is done with the command "sudo ./configure --arch = armel --target - os = linux --enable - gpl --enable - omx --enable - omx - rpi --enable - nonfree" (Figure 2.8).

```
pi@raspberrypi:~/usr/build/FFmpeg$ sudo ./configure --arch=armel --target-os=linux --enable-gpl --enable-omx --enable-omx-rpi --enable-nonfree
install prefix                /usr/local
source path                   .
C compiler                    gcc
C library                      glibc
ARCH                          arm (armv6)
big-endian                    no
runtime cpu detection         yes
ARMv5TE enabled               yes
ARMv6 enabled                 yes
ARMv6T2 enabled              yes
VFP enabled                   yes
NEON enabled                  yes
THUMB enabled                 no
debug symbols                 yes
strip symbols                 yes
optimize for size             no
optimizations                 yes
static                        yes
shared                        no
postprocessing support        yes
network support               yes
threading support             pthreads
safe bitstream reader         yes
texi2html enabled            no
perl enabled                  yes
pod2man enabled              yes
makeinfo enabled             no
makeinfo supports HTML      no
```

Figure 2.8: FFmpeg configuration.

Once the configuration is complete, FFmpeg is now ready to be compiled and the compile operation is started with the command "make -j4" (Figure 2.9). The "-j4" argument is necessary to allow the processor to compile using its all 4 cores. With this command, compile time is reduced up to 4 times, but it still may take a long time. After completing the compilation process, all the libraries have now been compiled and are now ready to be installed on the system. It can be seen that the size of the file now exceeds 800 MB when looked through the command "du -sh /usr/build/FFmpeg". To do a normal installation on Linux, it is needed to go to the folder that is compiled and type the command "make install". The compiled program will be automatically installed in the necessary system files, but if the installed program is later wished to be deleted or modified, some problems may be encountered, and it may not be deleted. By using "checkinstall" command instead of "make install" command, "deb" package can be created and installed in such a way that it can be easily removed with the command "dpkg -r ffmpeg" when it needs to be deleted in the future and possible deletion problems will be avoided.

```

pi@raspberrypi:/usr/build/FFmpeg $ sudo make -j4
GEN      libavutil/libavutil.version
GEN      libswscale/libswscale.version
GEN      libswresample/libswresample.version
GEN      libpostproc/libpostproc.version
GEN      libavcodec/libavcodec.version
GEN      libavformat/libavformat.version
GEN      libavfilter/libavfilter.version
GEN      libavdevice/libavdevice.version
CC      libavdevice/alldevices.o
CC      libavdevice/avdevice.o
CC      libavdevice/fbdev_common.o
CC      libavdevice/fbdev_dec.o
libavdevice/avdevice.c: In function 'device_next':
libavdevice/avdevice.c:88:13: warning: 'av_oformat_next' is deprecated [-Wdeprecated-declarations]
     if (!(prev = av_oformat_next(prev)))
     ^~
In file included from libavdevice/avdevice.h:51:0,
                  from libavdevice/avdevice.c:23:
./libavformat/avformat.h:2088:17: note: declared here
AVOutputFormat *av_oformat_next(const AVOutputFormat *f);
                  ^~
libavdevice/avdevice.c:92:13: warning: 'av_iformat_next' is deprecated [-Wdeprecated-declarations]
     if (!(prev = av_iformat_next(prev)))
     ^~
In file included from libavdevice/avdevice.h:51:0,
                  from libavdevice/avdevice.c:23:
./libavformat/avformat.h:2090:17: note: declared here
AVInputFormat *av_iformat_next(const AVInputFormat *f);
                  ^~
CC      libavdevice/fbdev_enc.o
CC      libavdevice/lavfi.o
CC      libavdevice/oss.o
CC      libavdevice/oss_dec.o

```

Figure 2.9: FFmpeg compile screenshot.

In order to create a deb package, the “checkinstall” command needs to be used via the terminal in the program directory where the "deb" file is to be created. If the “checkinstall” package is not available on the Linux system, it can be installed on the Linux system with the command "apt-get install checkinstall" (Figure 2.10).

```

pi@raspberrypi:/usr/build/nginx-1.10.3 $ sudo apt-get install checkinstall
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  gettext
The following NEW packages will be installed:
  checkinstall
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 134 kB of archives.
After this operation, 540 kB of additional disk space will be used.
Get:1 http://mirrors.linux-bg.org/raspbian/raspbian stretch/main armhf checkinstall armhf 1.6.2-4 [134 kB]
Fetched 134 kB in 2s (50.5 kB/s)
Selecting previously unselected package checkinstall.
(Reading database ... 45531 files and directories currently installed.)
Preparing to unpack ../checkinstall_1.6.2-4_armhf.deb ...
Unpacking checkinstall (1.6.2-4) ...
Processing triggers for man-db (2.7.6.1-2) ...
Setting up checkinstall (1.6.2-4) ...
pi@raspberrypi:/usr/build/nginx-1.10.3 $

```

Figure 2.10: The installation of the checkinstall package.

An example of the installation process with the checkinstall package is shown in Figure 2.11. After creating a Deb package and starting "checkinstall" command to start the installation process, the user will be asked to enter a description of the package. Figure 2.11 states that it is an FFmpeg version which is specially configured and compiled for Raspberry Pi by entering "ffmpeg for rpi".

```
pi@raspberrypi:/usr/build/FFmpeg $ sudo checkinstall
checkinstall 1.6.2, Copyright 2009 Felipe Eduardo Sanchez Diaz Duran
This software is released under the GNU GPL.

The package documentation directory ./doc-pak does not exist.
Should I create a default set of package docs? [y]: y

Preparing package documentation...OK

Please write a description for the package.
End your description with an empty line or EOF.
>> ffmpeg for rpi
>>
```

Figure 2.11: Entering a description for the deb package via checkinstall.

In the next step, the automatically generated package information is displayed. If the user wants to change the information, the user can change it by entering that information as a number value (Figure 2.12).

```
*****
*** Debian package creation selected ***
*****

*** Warning: The package name "FFmpeg" contains upper case
*** Warning: letters. dpkg might not like that so I changed
*** Warning: them to lower case.

This package will be built according to these values:

0 - Maintainer: [ root@raspberrypi ]
1 - Summary: [ ffmpeg for rpi ]
2 - Name: [ ffmpeg ]
3 - Version: [ 20180726 ]
4 - Release: [ 1 ]
5 - License: [ GPL ]
6 - Group: [ checkinstall ]
7 - Architecture: [ armhf ]
8 - Source location: [ FFmpeg ]
9 - Alternate source location: [ ]
10 - Requires: [ ]
11 - Provides: [ ffmpeg ]
12 - Conflicts: [ ]
13 - Replaces: [ ]

Enter a number to change any of them or press ENTER to continue:

Installing with make install...
```

Figure 2.12: Sample deb package description.

After the editing process is finished, when the enter key is pressed without entering the number, a package with the extension "deb" for ffmpeg is created and the installation process is started through this package. After the process is completed, the created package will have successfully been installed on the system. The result

screen will be as seen in Figure 2.13 and FFmpeg will be ready to use.

```
*****
Done. The new package has been installed and saved to
/usr/build/FFmpeg/ffmpeg_20180726-1_armhf.deb
You can remove it from your system anytime using:
    dpkg -r ffmpeg
*****
pi@raspberrypi: /usr/build/FFmpeg $
```

Figure 2.13: Creating a checkinstall package and completion of the installation process.

2.5.2 Compiling and installing Nginx with RTMP support

Nginx is a web server developed by Russian software engineer Igor Sysoev, developed as a lightweight, stable and fast mail client that is optimized for all servers. Due to the structure of Nginx, extra modules cannot be installed after installation. The modules that need to be loaded with Nginx needs to be specified in the configuration before being compiled with Nginx's source code. In this section, how Nginx will be compiled and installed with RTMP support will be explained step by step.

First, "cd /usr/build" command should be entered in the terminal to go to the "build" folder. It is then possible to download the Nginx source code for the version specified by the command "wget daily and stable version number>.tar.gz" to the build folder. The source code for "nginx-1.10.3.tar.gz" version is downloaded with the command "wget http://nginx.org/download/nginx-1.10.3.tar.gz" in the executed project. These source codes are compressed in "gzip" format. The command "tar xzf nginx-1.10.3.tar.gz" should be used to decompress the source code from the compressed file (Figure 2.14). Nginx is compilable after the file "tar xzf" is decompressed, but the Rtmp module is missing.

The RTMP module is downloaded to "/usr/build" directory with the command "sudo git clone git://github.com/arut/nginx-rtmp-module.git". After this step, the command "cd /usr/build/nginx-1.10.3/" is entered into the terminal, which changes the directory to where nginx source code is hosted. From this point forward, the

nginx server can be configured to be compilable with rtmp support. The configuration is done using the following command line: “./configure --prefix=/var/www_nginx --sbin-path=/usr/sbin/nginx --conf-path=/etc/nginx/nginx.conf --pid-path=/var/run/nginx.pid --error-log-path=/var/log/nginx/error.log --http-log-path=/var/log/nginx/access.log --with-http_ssl_module --without-http_proxy_module --add-module=../nginx-rtmp-module”.

```

pi@raspberrypi:~/usr/build $ sudo wget http://nginx.org/download/nginx-1.10.3.tar.gz
--2018-07-26 19:20:25-- http://nginx.org/download/nginx-1.10.3.tar.gz
Resolving nginx.org (nginx.org)... 95.211.80.227, 206.251.255.63, 2001:1af8:4060:a004:21::e3, ...
Connecting to nginx.org (nginx.org)|95.211.80.227|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 911509 (890K) [application/octet-stream]
Saving to: 'nginx-1.10.3.tar.gz'

nginx-1.10.3.tar.gz          100%[=====]
2018-07-26 19:20:28 (323 KB/s) - 'nginx-1.10.3.tar.gz' saved [911509/911509]

pi@raspberrypi:~/usr/build $ sudo tar xzf nginx-1.10.3.tar.gz
pi@raspberrypi:~/usr/build $ cd
FFmpeg/      nginx-1.10.3/      nginx-rtmp-module/
pi@raspberrypi:~/usr/build $ cd nginx-1.10.3/
pi@raspberrypi:~/usr/build/nginx-1.10.3 $ ./configure --prefix=/var/www --sbin-path=/usr/sbin/nginx --conf-path=
var/log/nginx/error.log --http-log-path=/var/log/nginx/access.log --with-http_ssl_module --without-http_proxy_mo

```

Figure 2.14: Example of downloading Nginx, extracting it from compressed file and configuring it.

There are some points to note in this configuration. With the use of the “--add-module=../nginx-rtmp-module” argument, the folder named "nginx-rtmp-module" is included in the configuration which is the upper folder of the folder where “configure” command is run.

With the use of the “--prefix=/var/www_nginx” argument, the use of /var/www/” as a public folder both with the nginx server and the Apache server, which is to be set-up later, is prevented by specifying this argument. Nginx will use the “/var/www_nginx/” folder for the web server. When the Nginx configuration starts, a screen output will be seen as shown in Figure 2.15. When the configuration is complete, a screen output as shown in Figure 2.16 will be displayed.

```

pi@raspberrypi:~/usr/build/nginx-1.10.3 $ sudo ./configure --prefix=/var/www --sbin-path=/usr/sbin/nginx --conf-path=
th=/var/log/nginx/error.log --http-log-path=/var/log/nginx/access.log --with-http_ssl_module --without-http_proxy_mo
checking for OS
+ Linux 4.14.50-v7+ armv7l
checking for C compiler ... found
+ using GNU C compiler
+ gcc version: 6.3.0 20170516 (Raspbian 6.3.0-18+rpil+deb9u1)
checking for gcc -pipe switch ... found
checking for -Wl,-E switch ... found
checking for gcc builtin atomic operations ... found
checking for C99 variadic macros ... found
checking for gcc variadic macros ... found
checking for gcc builtin 64 bit byteswap ... found

```

Figure 2.15: Sample output for Nginx configuration start.

```
Configuration summary
+ using system PCRE library
+ using system OpenSSL library
+ md5: using OpenSSL library
+ sha1: using OpenSSL library
+ using system zlib library

nginx path prefix: "/var/www"
nginx binary file: "/usr/sbin/nginx"
nginx modules path: "/var/www/modules"
nginx configuration prefix: "/etc/nginx"
nginx configuration file: "/etc/nginx/nginx.conf"
nginx pid file: "/var/run/nginx.pid"
nginx error log file: "/var/log/nginx/error.log"
nginx http access log file: "/var/log/nginx/access.log"
nginx http client request body temporary files: "client_body_temp"
nginx http fastcgi temporary files: "fastcgi_temp"
nginx http uwsgi temporary files: "uwsgi_temp"
nginx http scgi temporary files: "scgi_temp"

pi@raspberrypi:/usr/build/nginx-1.10.3 $
```

Figure 2.16: Sample output for Nginx configuration end.

Completed Nginx codes are now ready to be compiled. The compilation is started by typing the command "make -j4" in the folder where the nginx source files are located, as was seen in FFmpeg compilation before. The screen to be seen when compile is started is shown in Figure 2.17. The compilation operation of Nginx is shorter than FFmpeg compile operation.

```
pi@raspberrypi:/usr/build/nginx-1.10.3 $ sudo make -j4
make -f objs/Makefile
make[1]: Entering directory '/usr/build/nginx-1.10.3'
cc -c -pipe -O -W -Wall -Wpointer-arith -Wno-unused -Werror -g -I src/core -I src/event -I src/event/modules -I src/os/unix -I ../nginx-rtmp-module -I objs \
-o objs/src/core/nginx.o \
src/core/nginx.c
cc -c -pipe -O -W -Wall -Wpointer-arith -Wno-unused -Werror -g -I src/core -I src/event -I src/event/modules -I src/os/unix -I ../nginx-rtmp-module -I objs \
-o objs/src/core/nginx_log.o \
src/core/nginx_log.c
cc -c -pipe -O -W -Wall -Wpointer-arith -Wno-unused -Werror -g -I src/core -I src/event -I src/event/modules -I src/os/unix -I ../nginx-rtmp-module -I objs \
-o objs/src/core/nginx_palloc.o \
src/core/nginx_palloc.c
cc -c -pipe -O -W -Wall -Wpointer-arith -Wno-unused -Werror -g -I src/core -I src/event -I src/event/modules -I src/os/unix -I ../nginx-rtmp-module -I objs \
```

Figure 2.17: The start of the Nginx compilation process.

The screen image to be encountered after the completion of the compilation process is shown in Figure 2.18. After reaching this point, "deb" package will be created and installed in the system by the "checkinstall" command.

```
objs/addon/nginx-rtmp-module/nginx_rtmp_proxy_protocol.o \
objs/addon/hls/nginx_rtmp_hls_module.o \
objs/addon/dash/nginx_rtmp_dash_module.o \
objs/addon/hls/nginx_rtmp_mpegts.o \
objs/addon/dash/nginx_rtmp_mp4.o \
objs/addon/nginx-rtmp-module/nginx_rtmp_stat_module.o \
objs/addon/nginx-rtmp-module/nginx_rtmp_control_module.o \
objs/nginx_modules.o \
-ldl -lpthread -lcrypt -lpcrc -lssl -lcrypto -ldl -lz \
-Wl,-E
make[1]: Leaving directory '/usr/build/nginx-1.10.3'
pi@raspberrypi:/usr/build/nginx-1.10.3 $
```

Figure 2.18: The end of Nginx compilation process.

The screen to be encountered when the "checkinstall" command is executed for package creation is as shown in Figure 2.19. In this screen, a description is requested for the package to be created. The "nginx with rtmp server" description has been entered which indicates that it is different from the normal nginx server and contains the rtmp server support.

```
pi@raspberrypi:/usr/build/nginx-1.10.3 $ sudo checkinstall
checkinstall 1.6.2, Copyright 2009 Felipe Eduardo Sanchez Diaz Duran
This software is released under the GNU GPL.

The package documentation directory ./doc-pak does not exist.
Should I create a default set of package docs? [y]: y

Preparing package documentation...OK

Please write a description for the package.
End your description with an empty line or EOF.
>> nginx with rtmp server
```

Figure 2.19: Entering a description via checkinstall for the deb package for Nginx.

When the Enter key is pressed and passed to the next screen, the checkinstall software will automatically generate certain packet information and display a screen that allows the user to change what the user wants to change (Figure 2.20). If the user needs to make a change on this screen, enter the number of the option the user wants to change and press the enter key and change it. If no change is made, the package creation and installation process are started by pressing the enter key without any number key. If the package creation and installation process are completed successfully, the screen output in Figure 2.21. Now nginx has been successfully installed and ready for use on the local system. Access the "cd /var/www_nginx/html" folder and find the "index.html" sample page here. This page can be changed as desired. Nginx does not work at this stage.

In order to be able to run Nginx, the configuration file of the Nginx server will be modified first, and this change will make it possible to receive the rtmp broadcast that is made using FFmpeg and Raspivid and to re-broadcast it as hls. Before compilation, Nginx's configuration file is placed in the configuration file directory specified by the argument "--conf-path=/etc/nginx/nginx.conf".

```

pi@raspberrypi:/usr/build/nginx-1.10.3 $ sudo checkinstall
checkinstall 1.6.2, Copyright 2009 Felipe Eduardo Sanchez Diaz Duran
This software is released under the GNU GPL.

The package documentation directory ./doc-pak does not exist.
Should I create a default set of package docs? [y]: y

Preparing package documentation...OK

Please write a description for the package.
End your description with an empty line or EOF.
>> nginx with rtmp server
>>

*****
**** Debian package creation selected ****
*****

This package will be built according to these values:

0 - Maintainer: [ root@raspberrypi ]
1 - Summary: [ nginx with rtmp server ]
2 - Name: [ nginx ]
3 - Version: [ 1.10.3 ]
4 - Release: [ 1 ]
5 - License: [ GPL ]
6 - Group: [ checkinstall ]
7 - Architecture: [ armhf ]
8 - Source location: [ nginx-1.10.3 ]
9 - Alternate source location: [ ]
10 - Requires: [ ]
11 - Provides: [ nginx ]
12 - Conflicts: [ ]
13 - Replaces: [ ]

Enter a number to change any of them or press ENTER to continue: █

```

Figure 2.20: The description of the deb package for Nginx.

```

*****
Done. The new package has been installed and saved to
/usr/build/nginx-1.10.3/nginx_1.10.3-1_armhf.deb
You can remove it from your system anytime using:
    dpkg -r nginx
*****
pi@raspberrypi:/usr/build/nginx-1.10.3 $ █

```

Figure 2.21: Completion of package creation and installation for Nginx with the checkinstall.

Enter "nano /etc/nginx/nginx.conf" to open the "conf" file with the nano text editing program and change it to listen from port 81 first as shown in Figure 2.22, under "Server {" configuration. Then the configuration in Figure 2.23 is added for the

folder "hls" under "Server {" configuration. Then, Figure 2.24 configuration for rtmp server is added to the last part of the same config file, and the file is saved.

```
#gzip on;

server {
    listen      81;
    server_name localhost;

    #charset koi8-r;

    #access_log logs/host.access.log main;

    location / {
        root    html;
        index  index.php index.html index.htm;
    }

    #error_page 404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root    html;
    }
}
```

Figure 2.22: Nginx server configuration.

```
location /hls {
    # Disable cache
    add_header Cache-Control no-cache;

    # CORS setup
    add_header 'Access-Control-Allow-Origin' '*';
    add_header 'Access-Control-Expose-Headers' 'Content-Length';

    # allow CORS preflight requests
    if ($request_method = 'OPTIONS') {
        add_header 'Access-Control-Allow-Origin' '*';
        add_header 'Access-Control-Max-Age' 1728000;
        add_header 'Content-Type' 'text/plain charset=UTF-8';
        add_header 'Content-Length' 0;
        return 204;
    }

    types {
        application/vnd.apple.mpegurl m3u8;
        video/mp2t ts;
    }

    root /var/www/nginx/html;
}
```

Figure 2.23: Configuration of Nginx for hls broadcast.

```

rtmp {
    server {
        listen 1935;
        chunk_size 2048;

        application live {
            live on;
            record off;
            hls on;
            hls_path /var/www/nginx/html/hls;
        }
        application show {
            live on;
            pull rtmp://0.0.0.0:1935/live/stream live=1 name=stream;
            # to change the local stream name use this syntax: ... live=1 name=ch3;

            # other directives...
            # hls_...
        }
    }
}

```

Figure 2.24: Nginx configuration for running rtmp server.

In the last step, a service for Nginx needs to be created to enable Raspberry Pi to start it automatically after every boot. The Linux operating system service files are stored under the "/lib/systemd/system" directory. The command "nano /lib/systemd/system/nginx.service" will create a file named nginx.service and will allow entering the parameters shown in Figure 2.25. After the parameters shown in Figure 2.25 are entered and saved, the service files must be reloaded with the command "systemctl daemon-reload".

```

[Unit]
Description=The NGINX HTTP and reverse proxy server
After=syslog.target network.target remote-fs.target nss-lookup.target

[Service]
Type=forking
PIDfile=/run/nginx.pid
ExecStartPre=/usr/sbin/nginx -t
ExecStart=/usr/sbin/nginx
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s QUIT $MAINPID
PrivateTmp=true

[Install]
WantedBy=multi-user.target

```

Figure 2.25: nginx.service file content.

After this process is done, "sudo service nginx start" command starts Nginx and "systemctl enable nginx.service" command is entered to enable Nginx to start automatically every boot process. Now Nginx is able to receive rtmp broadcast and can rebroadcast it as "hls" for use on web pages. After this process is done, "sudo

service nginx start" command starts Nginx and "systemctl enable nginx.service" command is entered to enable Nginx to start automatically every boot process. Now Nginx is able to receive rtmp broadcast and can rebroadcast it as "hls" for use on web pages.

If the system starts recording at this point, the hls files will be attempted to be written on the SD card. As a result, this would affect the Raspberry Pi performance in a negative way significantly. To avoid this, it is necessary to mount the hls directory as a temporary directory to be saved on the ram. Mounting can be done with the command: `sudo "/bin/mount -t tmpfs -o size=100m tmpfs /var/www_nginx/html/hls"`. This command will mount the hls directory on to the ram. This temporary folder on the ram will have a 100mb limit.

At this point, the video from the Raspberry Pi camera can now be sent to the rtmp server running Nginx and the hls broadcast can be started. To start the broadcast, typing "raspivid -o - -t 0 -w 1280 -h 1024 -fps 25 -b 4000000 -g 50 | ffmpeg -f h264 -i - -vcodec copy -strict experimental -f flv rtmp://0.0.0.0/live/stream " from the terminal would be enough. In order to run this script automatically on every boot process, it is needed to put the command in /etc/rc.local.

2.5.3 Apache server, PHP, MariaDB, phpMyAdmin and web page installation

Apache is an open source and free Web server program that is developed by the Apache Software Foundation. It played a key role in the expansion of the World Wide Web and also, it is the most common Web server on the Internet to date [27].

Apache is the web server that will host the web page of the developed project. The installation can be done easily from the package repository via the command "apt-get install apache2 -y". The screen image to be encountered during the installation step is shown in Figure 2.26.

After the Apache2 installation is complete, the packages that will supply the PHP support for Apache can be installed on the running system with the command "apt-get -y install php7.0 libapache2-mod-php7.0 php7.0-mysql php7.0-curl php7.0-gd php7.0-intl php-pear php-imagine php7.0-imap php7.0-mcrypt php-memcache php7.0-pspell php7.0-recode php7.0-sqlite3 php7.0-tidy php7.0-xmlrpc php7.0-xsl ".

```

pi@raspberrypi:~$ sudo apt-get -y install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.2-0 ssl-cert
Suggested packages:
www-browser apache2-doc apache2-suexec-pristine | apache2-suexec-custom openssl-blacklist
The following NEW packages will be installed:
apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.2-0 ssl-cert
0 upgraded, 10 newly installed, 0 to remove and 0 not upgraded.
Need to get 1,953 kB of archives.
After this operation, 6,275 kB of additional disk space will be used.
Get:1 http://mirrors.linux-bg.org/raspbian/raspbian stretch/main armhf libapr1 armhf 1.5.2-5 [79.8 kB]
Get:2 http://mirrors.linux-bg.org/raspbian/raspbian stretch/main armhf libaprutil1 armhf 1.5.4-3 [75.9 kB]
Get:3 http://mirrors.linux-bg.org/raspbian/raspbian stretch/main armhf libaprutil1-dbd-sqlite3 armhf 1.5.4-3 [17.9 kB]
Get:4 http://mirrors.linux-bg.org/raspbian/raspbian stretch/main armhf libaprutil1-ldap armhf 1.5.4-3 [16.9 kB]
Get:5 http://mirrors.linux-bg.org/raspbian/raspbian stretch/main armhf liblua5.2-0 armhf 5.2.4-1.1 [82.8 kB]
Get:6 http://mirrors.linux-bg.org/raspbian/raspbian stretch/main armhf apache2-bin armhf 2.4.25-3+deb9u5 [1,042 kB]
Get:7 http://mirrors.linux-bg.org/raspbian/raspbian stretch/main armhf apache2-utils armhf 2.4.25-3+deb9u5 [219 kB]
Get:8 http://mirrors.linux-bg.org/raspbian/raspbian stretch/main armhf apache2-data all 2.4.25-3+deb9u5 [162 kB]
Get:9 http://mirrors.linux-bg.org/raspbian/raspbian stretch/main armhf apache2 armhf 2.4.25-3+deb9u5 [236 kB]
Get:10 http://mirrors.linux-bg.org/raspbian/raspbian stretch/main armhf ssl-cert all 1.0.39 [20.8 kB]
Fetched 1,953 kB in 12s (155 kB/s)

```

Figure 2.26: Installation of Apache2.

MariaDB is the new name for the new version of MySQL. The installation of MariaDB is identical to the installation of MySQL server. Information such as sensor data, users, etc. will be stored on MariaDB [28]. The installation of MariaDB is done by typing "sudo apt-get -y install mariadb-server mariadb-client" on the terminal. When this command is sent through the terminal, the screen output will be as shown in Figure 2.27.

```

pi@raspberrypi:~$ sudo apt-get -y install mariadb-server mariadb-client
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
galera-3 gawk libaio1 libarchive13 libcgi-fast-perl libcgi-pm-perl libdbd-mysql-perl libdbi-perl libencode-locale-perl libfcgi
libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl libjemalloc liblwp-mediatypes-perl liblz0-2 lib
libterm-readkey-perl libtimedate-perl liburi-perl lsof mariadb-client-10.1 mariadb-client-core-10.1 mariadb-common mariadb-ser
Suggested packages:
gawk-doc lrzlp libclone-perl libldb-perl libnet-daemon-perl libsql-statement-perl libdata-dump-perl libipc-sharedcache-perl
The following NEW packages will be installed:
galera-3 gawk libaio1 libarchive13 libcgi-fast-perl libcgi-pm-perl libdbd-mysql-perl libdbi-perl libencode-locale-perl libfcgi
libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl libjemalloc liblwp-mediatypes-perl liblz0-2 lib
libterm-readkey-perl libtimedate-perl liburi-perl lsof mariadb-client mariadb-client-10.1 mariadb-client-core-10.1 mariadb-com
mariadb-server-core-10.1 mysql-common soeat
0 upgraded, 35 newly installed, 0 to remove and 0 not upgraded.
Need to get 23.7 MB of archives.
After this operation, 174 MB of additional disk space will be used.

```

Figure 2.27: Installation of MariaDB.

Phpmyadmin is a software that provides a visual web interface to manage databases maintained in MariaDB. With this software, it is possible to import or export SQL databases onto MariaDB and also make changes in tables of any database. This allows the actions that need to be executed on the database side to be done in an easy and visual way. To install PhpMyAdmin, the command "apt-get install phpmyadmin -y" needs to be run via the terminal (Figure 2.28). During the installation, the program will guide the user and complete the necessary operations.

The final action that needs to be taken to run phpMyAdmin after the completion of the installation is to add the line "Include /etc/phpmyadmin/apache.conf", which is the configuration file for Phpmyadmin that is prepared for Apache, in the

configuration of Apache2 after entering the command “nano /etc/apache2/apache2.conf”.

```
pi@raspberrypi:~/var/www/html$ sudo apt-get -y install phpmyadmin
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  dbconfig-common dbconfig-mysql javascript-common libjs-jquery libjs-sphinxdoc libjs-underscore libsip4 php-bz2 php-mbstring php-mysql php-php-gettext php-phpseclib php-tcpdf
  php-zip php7.0-bz2 php7.0-mbstring php7.0-zip
Suggested packages:
  php-libsodium php-gmp php5-imagick www-browser
Recommended packages:
  php5-gd php5-mcrypt
The following NEW packages will be installed:
  dbconfig-common dbconfig-mysql javascript-common libjs-jquery libjs-sphinxdoc libjs-underscore libsip4 php-bz2 php-mbstring php-mysql php-php-gettext php-phpseclib php-tcpdf
  php-zip php7.0-bz2 php7.0-mbstring php7.0-zip phpmyadmin
0 upgraded, 18 newly installed, 0 to remove and 0 not upgraded.
Need to get 19.4 MB of archives.
After this operation, 80.3 MB of additional disk space will be used.
Get:2 http://rasbian.raspberrypi.org/raspbian stretch/main armhf dbconfig-mysql all 2.0.0 [996 B]
Get:4 http://mirrors.linux-99.org/raspbian/raspbian stretch/main armhf dbconfig-common all 2.0.0 [617 kB]
Get:3 http://mirrors.linux-99.org/raspbian/raspbian stretch/main armhf javascript-common all 11.6.120 B
```

Figure 2.28: Installation of PhpMyAdmin.

After the installation is done, file permissions settings for / var / www and / var / www_nginx directories must be adjusted. As a result of this setting, users of "pi" and "www-data" will be given access to these folders and all files in the folder will be given the authority of this group and consequently will be able to access the folders named "www-data" and "pi" users. This is done by processing the command sequence shown in Figure 2.29.

```
pi@raspberrypi:~$ sudo chown -R pi:www-data /var/www_nginx/
pi@raspberrypi:~$ chgrp -R www-data /var/www
pi@raspberrypi:~$ chgrp -R www-data /var/www_nginx/
pi@raspberrypi:~$ chmod -R g+rwx /var/www
pi@raspberrypi:~$ chmod -R g+rwx /var/www_nginx/
```

Figure 2.29: User access permissions for folders.

Once this is done, the web page code which is developed with PHP and AJAX is loaded into the “/var/www/html” directory via "Filezilla" software. Then the database is dumped from the server where the code was developed. It is imported via phpMyAdmin as shown in Figure 2.30. The next screen image, which is taken after the import, can be seen in Figure 2.31.

Once the database is imported, the database username and password, which provides the connection between the web page and the database, is saved in the Config.php file. Then, any computer connected to the same network can access the site by entering the IP address of Raspberry Pi on the web page of the browser (Figure 2.32).

Importing into the current server

File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
 A compressed file's name must end in **.[format].[compression]**. Example: **.sql.zip**

Browse your computer: Dump20180724.sql (Max: 2,048KiB)

You may also drag and drop a file on any page.

Character set of the file:

Partial import:

Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. (*break transactions.*)

Skip this number of queries (for SQL) starting from the first one:

Other options:

Enable foreign key checks

Format:

Console

Figure 2.30: Importing the SQL dump.

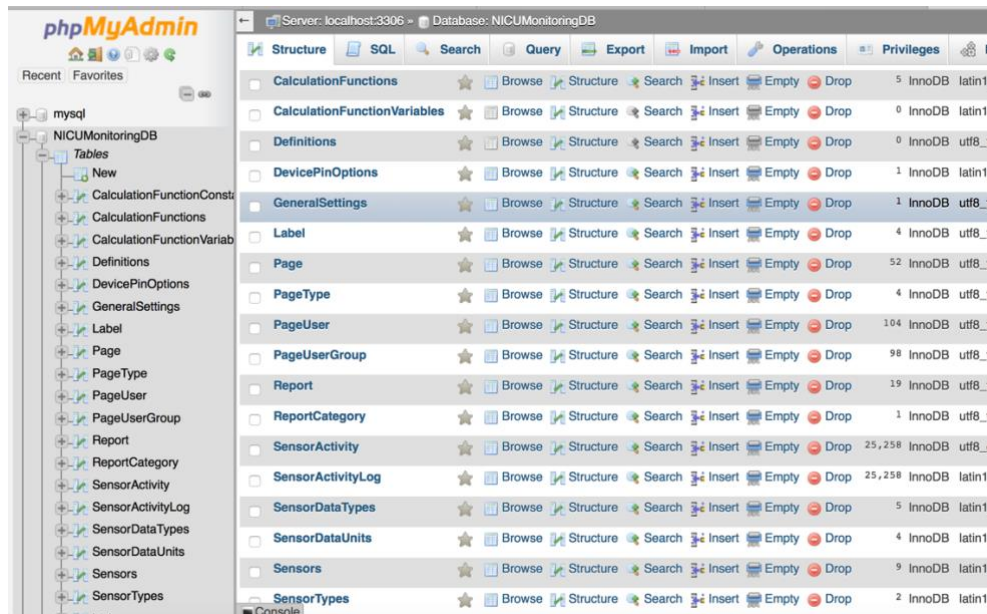


Figure 2.31: phpMyAdmin database screenshot.

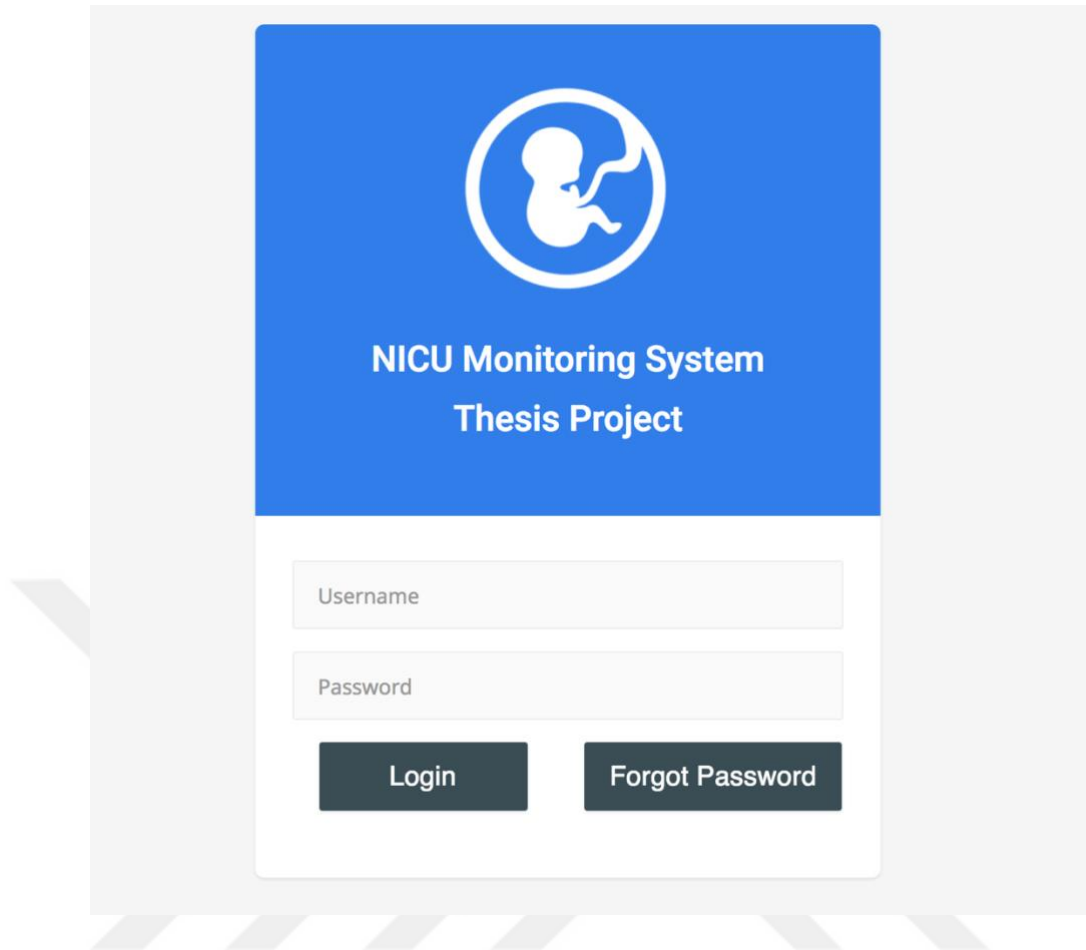


Figure 2.32: NICU Monitoring System login screen.

2.6 Python Scripts

The NICU Monitoring System project maintains a webservice page that works with the "GET" method to save sensor values in the "SensorActivity" table in the "NICUMonitoringDB" database that MariaDB contains. The path of this page is "/Var/www/html/NICUMonitoringProject/Services/SensorActivityWS/URL/index.php". The function on this page takes two variables. These variables are sensorId (sensor name) and sensorDataValue (read sensor value). A separate script has been prepared for each sensor type to read these values from the sensors and send them to the NICU Monitoring System site via webservice.

2.6.1 ADS1115 ADC python script

The ADS1115 uses the I2C protocol to read and send 16-bit, 4-channel analog data. In order to read data from this IC, the I2C interface of Raspberry Pi needs to be

activated. In order to enable the I2C interface, Raspberry settings should be changed, and to do this, the command "raspi-config" needs to be entered first. From the next screen "Interfacing Options" menu, which is indicated by number 5, needs to be selected. The menu can be seen in Figure 2.33. Next, the item specified as "P5 I2C" needs to be selected and finally, "YES" option is selected for the incoming query as shown in Figure 2.34. Raspberry Pi's I2C interface is now activated (Figure 2.35).

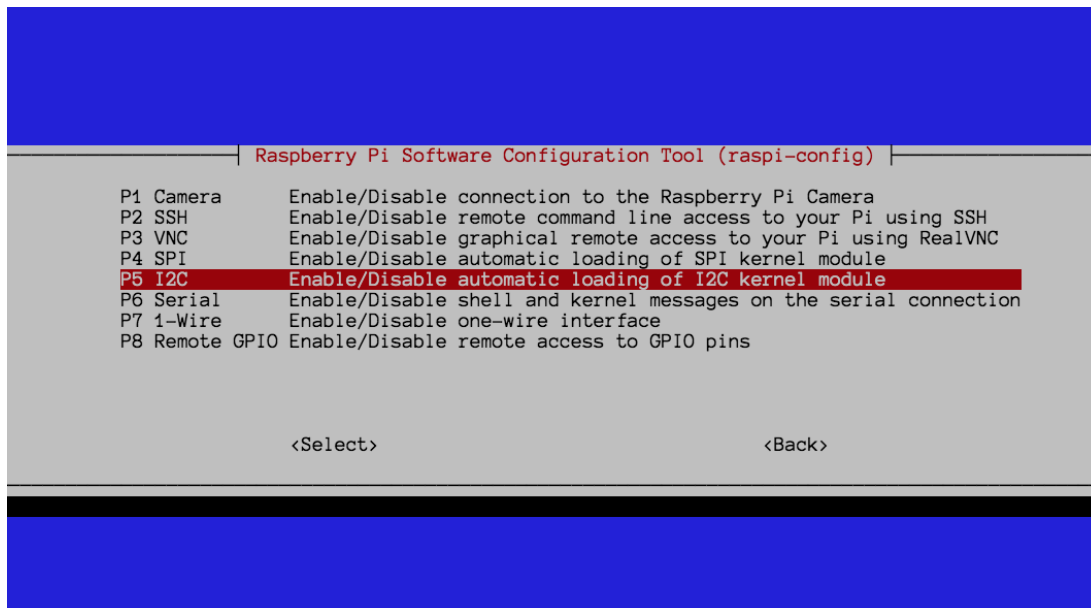


Figure 2.33: Interfacing options:I2C.

Once activated, the ICs that are communicating through the I2C protocol become visible at I2C addresses and also become communicable. The user can use "sudo i2cdetect -y 1" command to see the ICs and their addresses on the line. The output of this command is as seen in Figure 2.36.

The address of the ADS1115 IC is "48". If it is necessary to communicate with this IC, address 48 must be used. There are packages for the ADS1115 IC which are developed for Python. With the command "git clone https://github.com/adafruit/Adafruit_Python_ADS1x15.git", the package can be easily downloaded and installed onto the system [29].

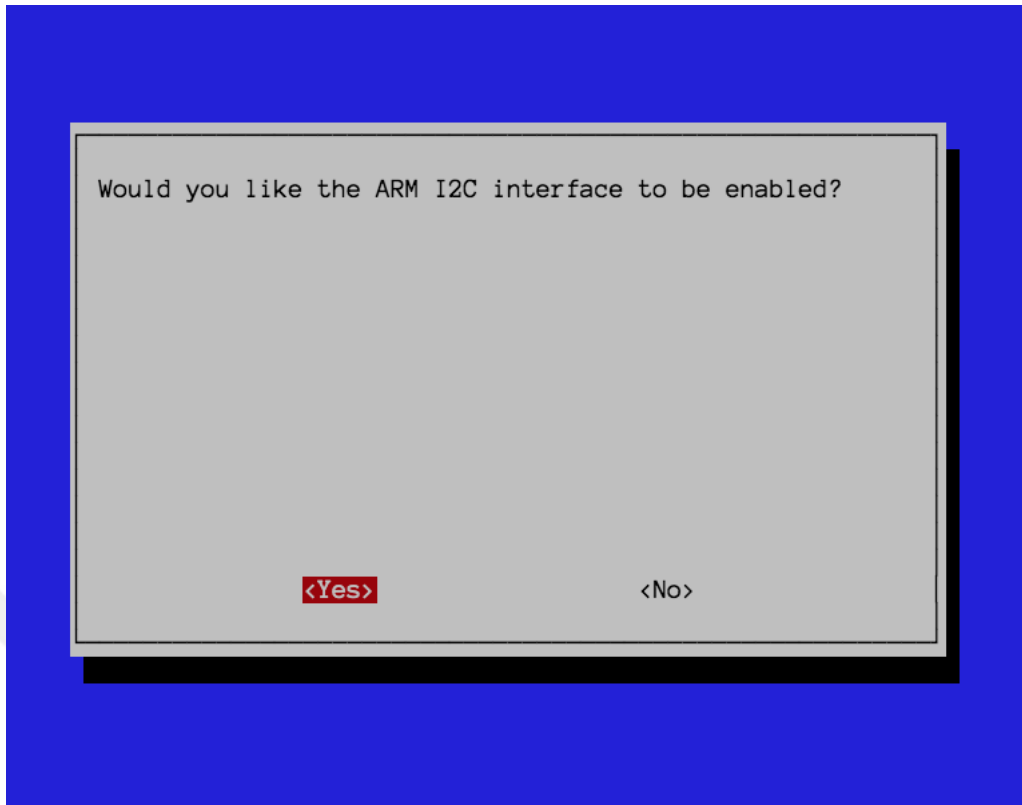


Figure 2.34: Enabling the I2C interface.



Figure 2.35: The prompt after enabling the I2C interface.

```

pi@raspberrypi:~ $ sudo i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40:  -- -- -- -- -- -- -- 48 -- -- -- -- -- -- --
50:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

```

Figure 2.36: The accessible addresses that are on the I2C data line.

For installation onto the system, the user may need to install "apt-get install python3-setuptools python3-pip" packages on the user's system. If the packages are already downloaded, they are loaded into the system by entering the downloaded Adafruit_Python_ADS1x15 folder and executing "python3 setup.py install" command. Then use the command "python3 ADS1115_READ.py" to run the python script for ADS1115 (Figure 2.37). The gain value in the code determines the voltage range to read.

```

import time
import urllib.request
import Adafruit_ADS1x15
time.sleep(30)
def get_service(deviceid,value):
    contents = urllib.request.urlopen("http://localhost/NICUMonitoringProject")
    return contents

adc = Adafruit_ADS1x15.ADS1115()
GAIN = 1
print('| {0:>6} | {1:>6} | {2:>6} | {3:>6} |'.format(*range(4)))
print('-' * 37)
while True:
    values = [0]*4
    for i in range(4):
        values[i] = adc.read_adc(i, gain=GAIN)
        get_service("ADS1115-A"+str(i),str(values[i]))
        time.sleep(0.5)
        print(str(values[i]))
    print('| {0:>6} | {1:>6} | {2:>6} | {3:>6} |'.format(*values))
    time.sleep(5)

```

Figure 2.37: ADS1115_READ.py file.

The script reads the analog channels every five seconds and sends each one separately to the web service.

2.6.2 DS18B20 and DS18S20 temperature sensors

The DS18B20 and DS18S20 communicate on a single line called the onewire. Data is received and sent via this line. Raspberry Pi supports the 1-wire protocol. In order to activate this communication interface, after entering "raspi-config" command, the user has to enter "Interfacing Options" menu which is indicated with number 5 on the following screen, select the option as shown in Figure 2.38 and then select "Yes" as shown in Figure 2.39. At the end of this operation, one-wire is activated and the screen output is as shown in Figure 2.40.

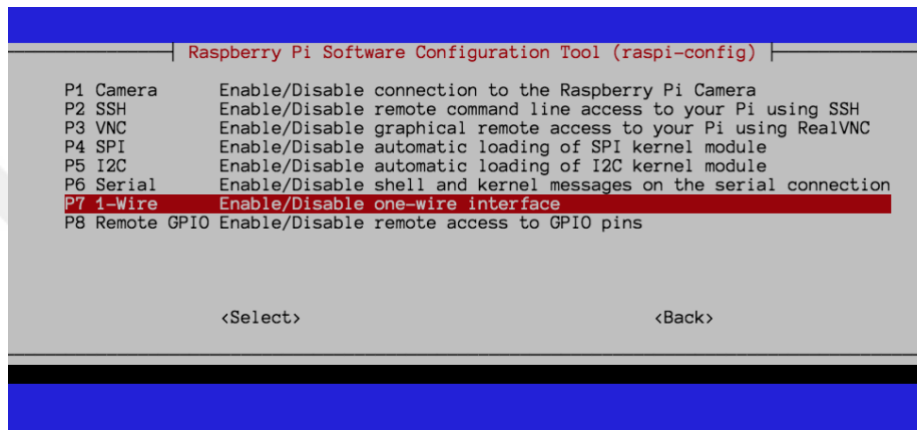


Figure 2.38: Interfacing options 1-Wire.

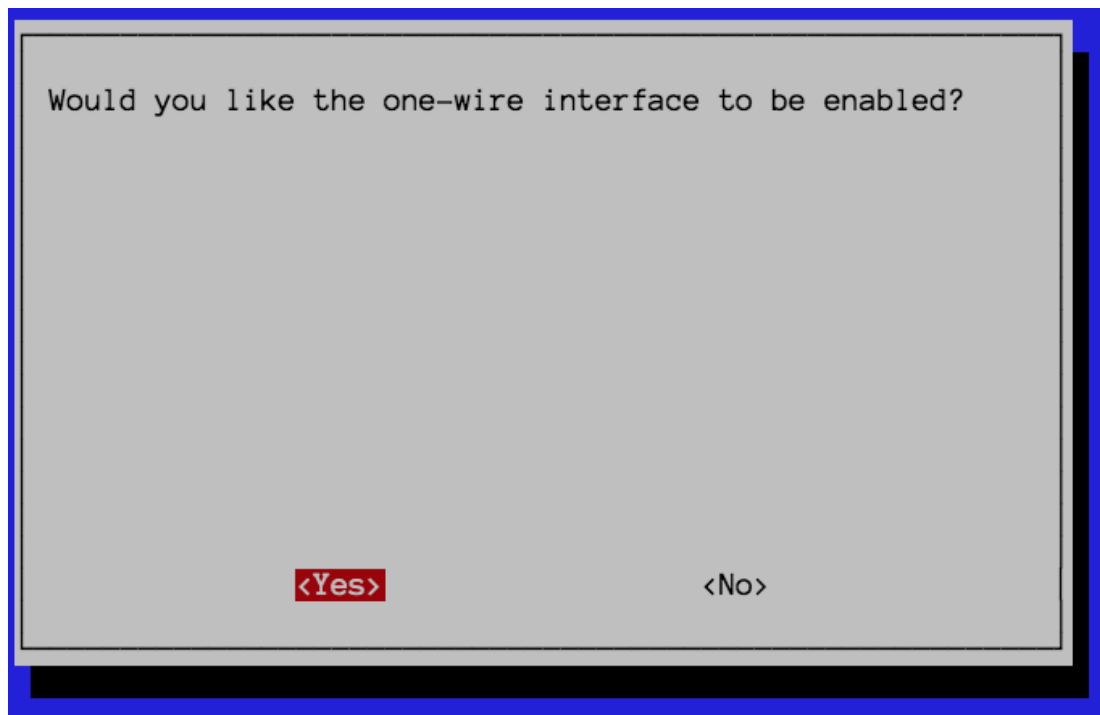


Figure 2.39: Enabling the 1-wire interface.



Figure 2.40: The prompt for the confirmation of one-wire being enabled.

Once activated, the sensors connected to the one-wire data line begin communicating. The example output of the command “ls /sys/bus/w1/devices” is shown in Figure 2.41. The sensors that start with “10-” and “28-” in this example output are the DS18B20 and DS18S20 sensors. They can easily be read by the command “tail /sys/bus/w1/devices/ 1000080218867d/w1_slave” (Figure 2.42).

```
pi@raspberrypi:~$ ls /sys/bus/w1/devices
10-00080218867d 10-000802189d4c 28-000006316ff2 w1_bus_master1
```

Figure 2.41: One-wire bus devices.

```
pi@raspberrypi:~$ tail /sys/bus/w1/devices/10-00080218867d/w1_slave
3c 00 4b 46 ff ff 09 10 2d : crc=2d YES
3c 00 4b 46 ff ff 09 10 2d t=30187
```

Figure 2.42: Sample output from reading sensors.

Each of the folders listed in the /devices folder belongs to a sensor. The folder names starting with “10-” and “28-” are unique identifiers of the sensors. The python script that is written takes a list of the folders in this folder, reads in the output of the “w1_slave” file in each folder, and after the string truncation, the resulting number is divided by 1000 to get the temperature value.

```

import os
import time

import urllib.request
time.sleep(30)
readtemp = 800,800
alarm_temp=20.0
os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'
prefixed = [filename for filename in os.listdir(base_dir) if( filename.starts

device_num=len(prefixed)
readtemp={}

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        temp_f = temp_c * 9.0 / 5.0 + 32.0
        return round(temp_c,2), temp_f

def get_service(deviceid,value):
    contents = urllib.request.urlopen("http://localhost/NICUMonitoringProject
    return contents

while True:
    readtemp.clear()
    prefixed = [filename for filename in os.listdir(base_dir) if( filename.st
    msg=""
    for counter, deviceid in enumerate(prefixed):
        try:
            print("device_id: " + deviceid)
            # print "device_name: " + filter(lambda person: person['SENSORID']
            print("counter: " + str(counter))
            device_folder = base_dir + deviceid
            device_file = device_folder + '/w1_slave'
            readtemp[deviceid]=read_temp()[0]
            print("temperature: " + str(readtemp[deviceid]))
            print("-----")
            get_service(str(deviceid),str(readtemp[deviceid]))
        except IOError:
            traceback.print_exc()
            pass
        except IndexError:
            traceback.print_exc()
            pass
        except:
            traceback.print_exc()
        else:
            pass
    time.sleep(10)

```

Figure 2.43: DS1820.py file.

The DS1820.py script, shown in Figure 2.43, performs sensor readings and sends values to the web service using the "GET" method.

2.6.3 SHT11 temperature and humidity sensor

SHT11 is a sensor developed by Sensirion, which provides high accuracy measurement of humidity and temperature. This sensor has its own data protocol. Packages written for Python can be found on Github.

```
import os
import time
import urllib.request
import traceback
import RPi.GPIO as GPIO
from pi_sht1x import SHT1x

DATA_PIN = 18
SCK_PIN = 17

def get_service(deviceid,value):
    contents = urllib.request.urlopen("http://localhost/NICUMonitoringProject0
    return contents

def temp_read():
    with SHT1x(DATA_PIN, SCK_PIN, gpio_mode=GPIO.BCM, vdd='3.5V', resolution=
        heater=False, otp_no_reload=False, crc_check=True) as sensor:

        temp = sensor.read_temperature()
        humidity = sensor.read_humidity(temp)
        #dewsensor.calculate_dew_point(temp, humidity)
        return {"humidity":humidity,"temperature":temp}

while True:
    msg=""
    get_hum_temp=temp_read()
    try:

        print("device_id: " + "sht11")
        print("temperature: " + str(get_hum_temp["temperature"]))
        print("humidity: " + str(get_hum_temp["humidity"]))
        print("-----")
        get_service(str("sht11-H"),str(get_hum_temp["humidity"]))
        get_service(str("sht11-T"),str(get_hum_temp["temperature"]))

    except IOError:
        traceback.print_exc()
        pass
    except IndexError:
        traceback.print_exc()
        pass
    except:
        traceback.print_exc()
    else:
        pass
    time.sleep(10)
```

Figure 2.44: SHT11.py file.

In order to read data from the sht11 sensor in the project, pi-sht1x package was installed on the system by the command "pip3 install git + https://github.com/drohm/pi-sht1x". After this step, a Python script can be written for SHT11. Figure 2.44 shows the Python script written for SHT11. The script reads the temperature and humidity values every ten seconds from the sensor and sends it to the site via web service.

2.7 The Configuration and Content of the Web Page

The web page has been designed to be as user-friendly as possible. In the upper section, there is an area to show instant data from the sensors. Sensor data is read as raw data with the help of Python scripts before it arrives in this section. Afterward, the project is transmitted to the web service section by the "GET" method. The raw data that is transmitted to the database is stored in the in the table called "SensorActivityLog" as raw data. A sample output is shown in Figure 2.45.

id	receivedSensorId	receivedSensorDataValue	requestTime
58476	5	30.88	2018-08-05 23:09:38
58477	13	13246	2018-08-05 23:09:43
58478	7	30.88	2018-08-05 23:09:51
58479	9	30.71	2018-08-05 23:09:53
58480	8	30.88	2018-08-05 23:09:55
58481	10	26276	2018-08-05 23:09:58
58482	11	23410	2018-08-05 23:10:00
58483	12	13144	2018-08-05 23:10:02
58484	13	13250	2018-08-05 23:10:04
58485	6	47.03	2018-08-05 23:10:06
58486	5	30.88	2018-08-05 23:10:07
58487	9	30.71	2018-08-05 23:10:10
58488	10	26256	2018-08-05 23:10:11
58489	7	30.88	2018-08-05 23:10:12
58490	11	23406	2018-08-05 23:10:16
58491	8	30.94	2018-08-05 23:10:17
58492	12	13142	2018-08-05 23:10:18
58493	13	13244	2018-08-05 23:10:21
58494	6	47.1	2018-08-05 23:10:25
58495	9	30.72	2018-08-05 23:10:29
58496	5	30.88	2018-08-05 23:10:31
58497	7	30.88	2018-08-05 23:10:32
58498	10	26264	2018-08-05 23:10:32

Figure 2.45: SensorActivityLog table.

The data received via the Python script and the web service is also recorded in the table named "SensorActivity" which is shown in Figure 2.47, after being calculated in accordance with the formulas of the function defined in the table named "CalculationFunctions" that is shown in Figure 2.46. The sensor and function that is going to be used depends on the type of sensor data and adjusted accordingly.

id	functionName	functionDescription	formulation
4	celsius	general celsius function for digital temperature s...	x
5	humidity	general function for digital humidity sensors	x
6	ADS1115 VOLT	ADS1115 VOLT CONVERT	(4.096/65535)*x

Figure 2.46: CalculationFunctions table.

The table with this information is set to the "Sensors" table shown in Figure 2.48. The table "Sensors" contains the name of the sensor, whether it is analogue or digital, what kind of data it has, the sensor name, the name of the sensor sent with the "GET" command, the alarm status value set which is set for the sensor and the alarm status. The database also contains pin outputs and pin names that enable connected devices to generate trigger signals for their operation under the "DevicePinOptions" table.

id	sensorid	activityValue	activityTime	usedCalculationFunctionId
58476	6	47.07	2018-08-05 23:09:49	5
58477	13	0.8278750	2018-08-05 23:09:49	6
58478	7	30.88	2018-08-05 23:09:53	4
58479	9	30.71	2018-08-05 23:09:54	4
58480	8	30.88	2018-08-05 23:09:56	4
58481	10	1.6422500	2018-08-05 23:09:58	6
58482	11	1.4631250	2018-08-05 23:10:01	6
58483	12	0.8215000	2018-08-05 23:10:02	6
58484	13	0.8281250	2018-08-05 23:10:04	6
58485	6	47.03	2018-08-05 23:10:08	5
58486	5	30.88	2018-08-05 23:10:10	4
58487	9	30.71	2018-08-05 23:10:12	4
58488	10	1.6410000	2018-08-05 23:10:14	6
58489	7	30.88	2018-08-05 23:10:14	4
58490	11	1.4628750	2018-08-05 23:10:17	6
58491	8	30.94	2018-08-05 23:10:19	4
58492	12	0.8213750	2018-08-05 23:10:19	6
58493	13	0.8277500	2018-08-05 23:10:26	6
58494	6	47.1	2018-08-05 23:10:26	5
58495	9	30.72	2018-08-05 23:10:29	4
58496	5	30.88	2018-08-05 23:10:31	4
58497	7	30.88	2018-08-05 23:10:32	4

Figure 2.47: SensorActivity table.

id	sensorTypeid	sensorNumber	sensorName	alarmRate	sensorDataTyped	alarmControlType
5	2	10-000802189d4c	PROBE1	30	4	1
6	2	sht11-H	Humidity	60	5	1
7	2	10-00080218867d	PROBE2	30	4	2
8	2	28-000006316ff2	PROBE3	20	4	2
9	2	sht11-T	TEMP-SHT	40	4	1
10	1	ADS1115-A0	AN0	3.2	6	1
11	1	ADS1115-A1	AN1	1.4	6	1
12	1	ADS1115-A2	AN2	3.2	6	2
13	1	ADS1115-A3	AN3	2.3	6	2

Figure 2.48: Sensors table.

2.8 Content and Configuration of the Web Page

The screen that is going to be seen, when the ip address which is associated with the dhcp server to the Raspberry Pi is entered, will be as shown in Figure 2.32. The screen that will be encountered when logged in to the system by entering the user information on this page is basically composed of 4 sections.

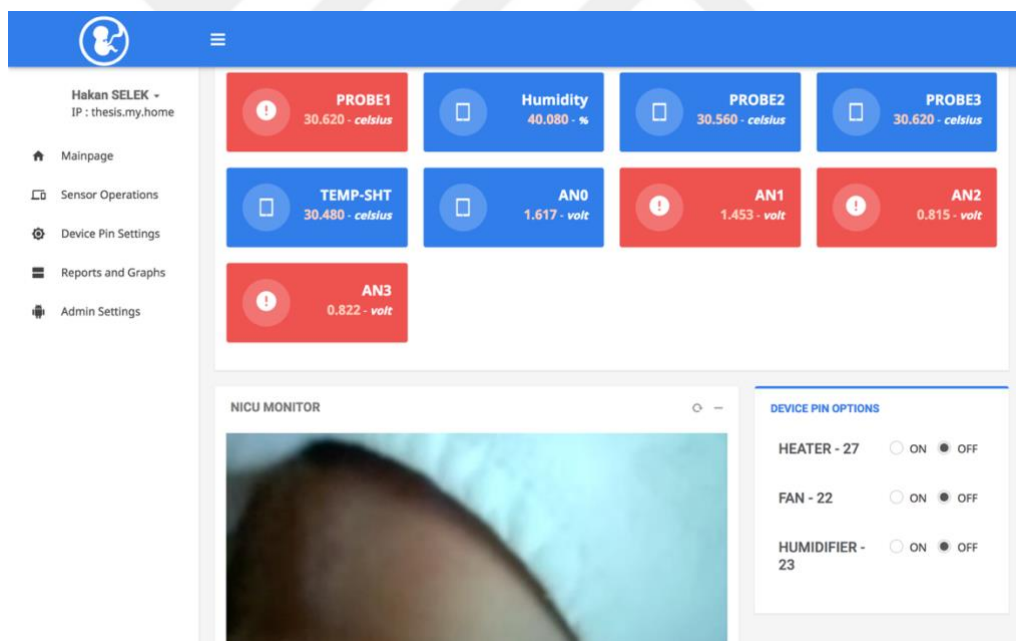


Figure 2.49: Screenshot of the main page.

These are the pin settings that provide the output control triggers as shown in Figure 2.49, the area responsible for displaying the sensor data and give an alarm when any sensor reaches the defined alarm value, and when not in an alarm state would show the live value of the sensors, a menu area for setting, screening, reporting and

checking all system related configurations, and an area for showing the live broadcast from the camera, which enables the incubator to be monitored.

In order for the sensor data read by the Python script to be sent to the web service of the project, the data should be added to the database tables in the system as shown in the examples in Figure 2.46 and Figure 2.48, after passing through the necessary pages and various operations.

2.8.1 Assigning trigger pins to the system

Controlling devices inside the incubator such as fan, humidifier, and heater remotely would be beneficial and would prevent the baby from waking up. For this purpose, a specific area on the right side of the screen which is called “DEVICE PIN OPTIONS” is designed for the control of all the available GPIO pins on the Raspberry Pi and it can be seen in Figure 2.49.

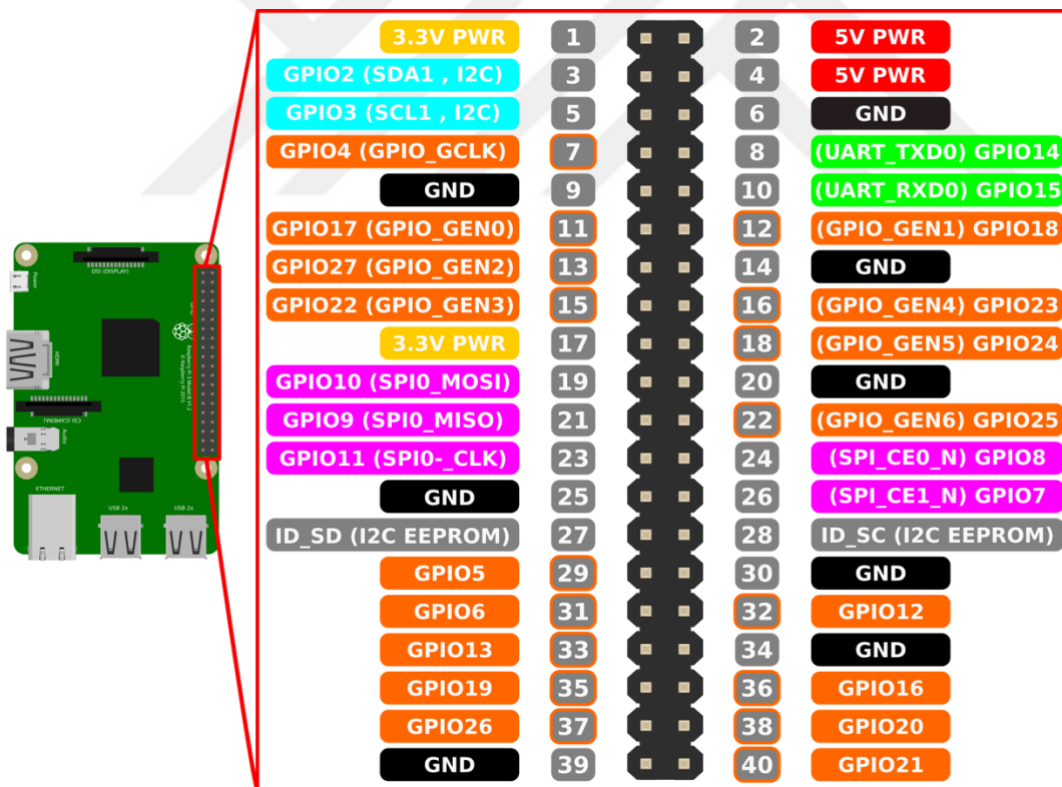


Figure 2.50: Raspberry Pi GPIO pin chart.

The devices mentioned here are shown in the format of “DEVICE NAME – GPIO ID in Use”. For example, for the HEATER, the pin which has the id 17 from the

GPIO pin chart and 11 in sequential order is used (Figure 2.50). If another device is needed to be defined, “Device Add Pin” option under the “Device Pin Settings” from the left side menu needs to be selected. In this section, the name of the device, the id of the GPIO pin number which is going to be triggered, and the default value of on or off in the case of an available connection is selected (Figure 2.51). When the new device is added, it is now added to the list of devices on the right under “DEVICE PIN OPTIONS” that can be seen in Figure 2.49. The added device can now be controlled through the assigned pin.

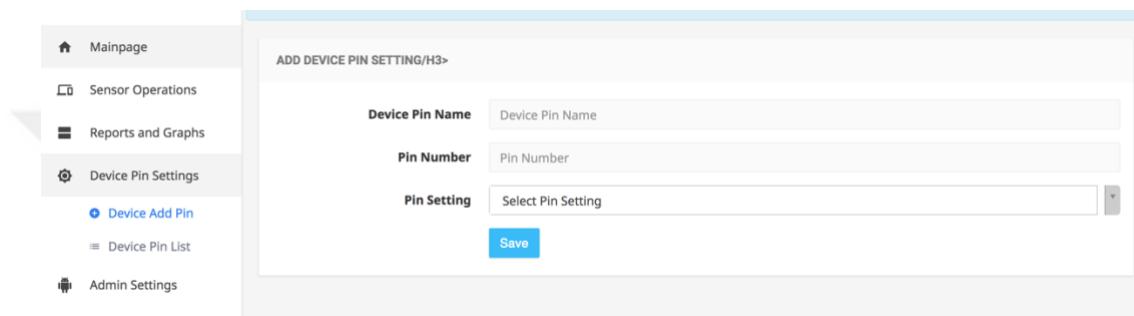


Figure 2.51: Device add pin page.

2.8.2 List of device-pin definition

In this page, the device-pin definitions defined to the system via the "Device Add Pin" page, as shown in Figure 2.51, are listed and can be exported to file formats such as PDF, EXCEL, and CSV (Figure 2.52).

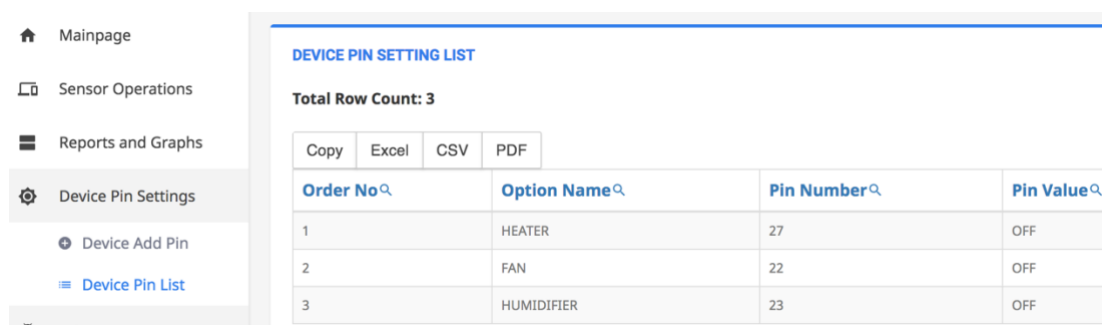


Figure 2.52: Device pin list page.

2.8.3 Reports and graphs page

From this page (Figure 2.53), the list of the values of the sensors that are registered in the system can be obtained or viewed as a graph.

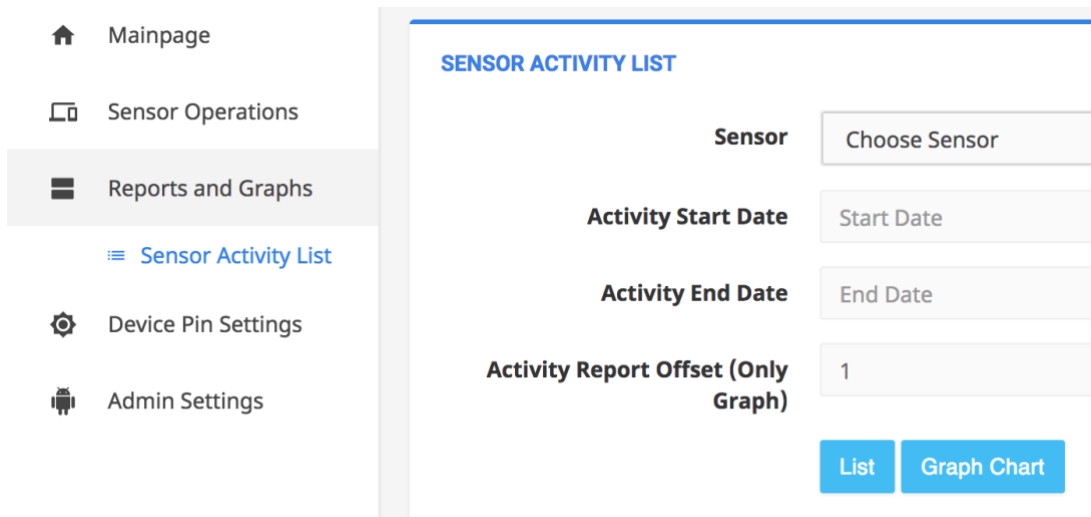


Figure 2.53: Reports and graphs page.

After selecting the sensor for which the list or graph is requested, the "Activity Start Date" and "Activity End Date" sections are used to filter the data received for those dates.

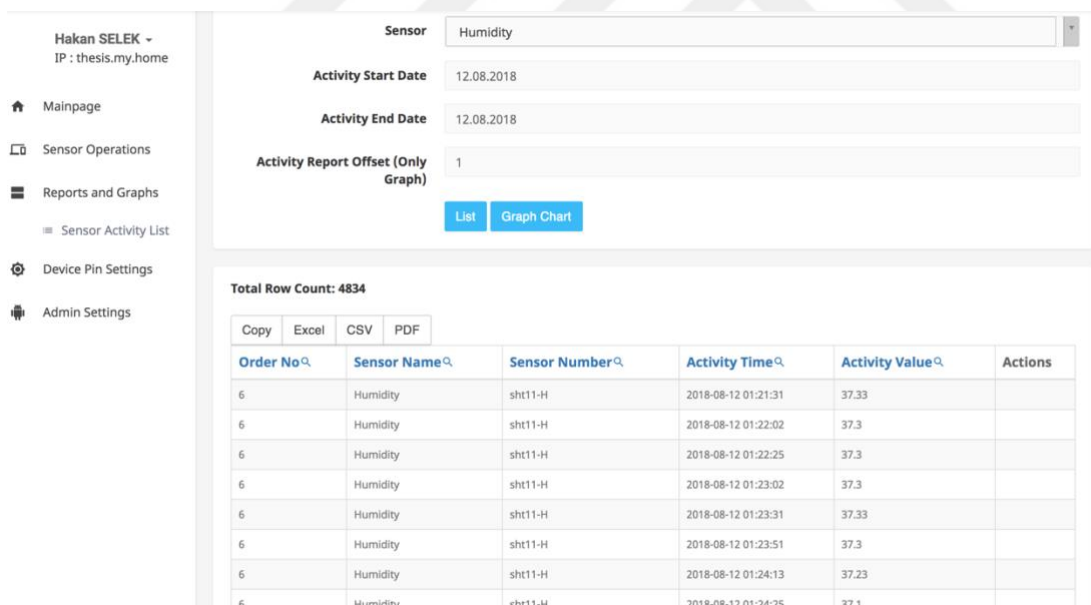


Figure 2.54: Sensor activity list screenshot.

Then, if only the data needs to be listed, the "List" Button is clicked. When system operation is complete, the selected sensor will list the data in the specified date range, as shown in Figure 2.54. If a graphic output is desired, "Activity Report Offset" part should also be specified. This field has a default value of 1. The

"Activity Report Offset" field sets the sampling amount. For example, if the user has 100 data and if the user enters the value 10 in this field, 1 will be taken from every 10 data and the user will have a total of 10 data. "Activity Report Offset" is set to 50 in the graphic page example, which can be seen in Figure 2.55. In this case, graphs were generated by taking 1 out of 50 samples. The generated graphic also contains the date information.



Figure 2.55: Sensor activity graph screenshot.

2.8.4 Add data unit page

In this page (Figure 2.56), the units of the value that is going to be output as a result of the function are defined. The unit that is defined on this page will be displayed in the sensors section of the main page. For example, if the pressure sensor is to be used, the resulting value for the function to be used must have a value such as "mmHg", "kPa" or "atm".

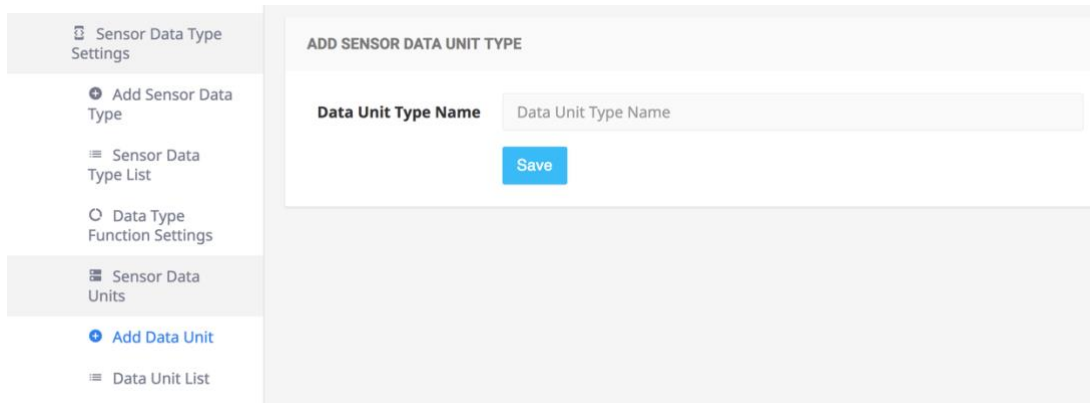


Figure 2.56: Add Data Unit Page screenshot.

2.8.5 Data unit list page

In this page, the unit types (Figure 2.57) which are defined to the system in the “Add Data Unit” section (Figure 2.56) are listed. Also, from this page, this data can be exported to PDF, EXCEL, CSV formats.

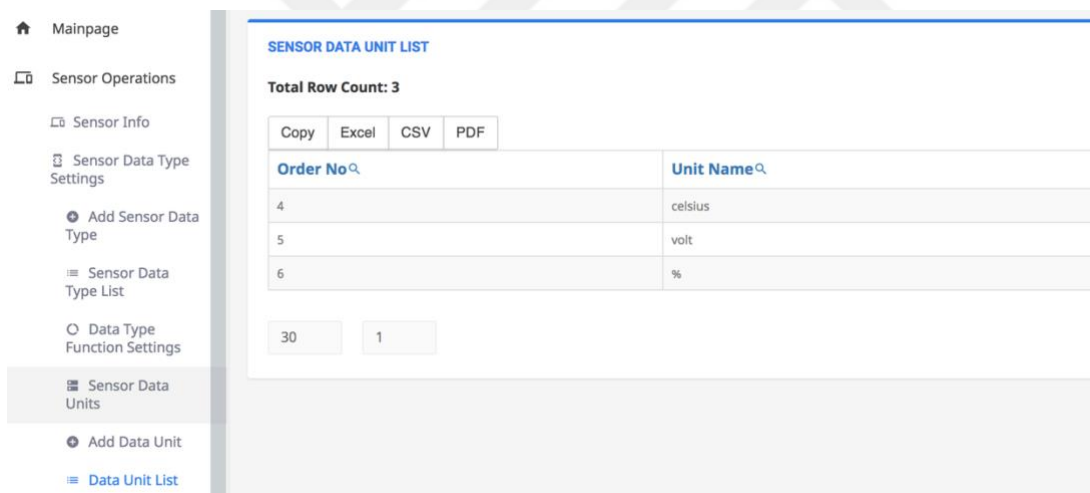


Figure 2.57: Data unit list page screenshot.

2.8.6 Add function page

In this page (Figure 2.58), the functions which will process the raw data are defined. For example, as mentioned before, the ADS1115 IC is an ADC IC with 16-bit resolution.

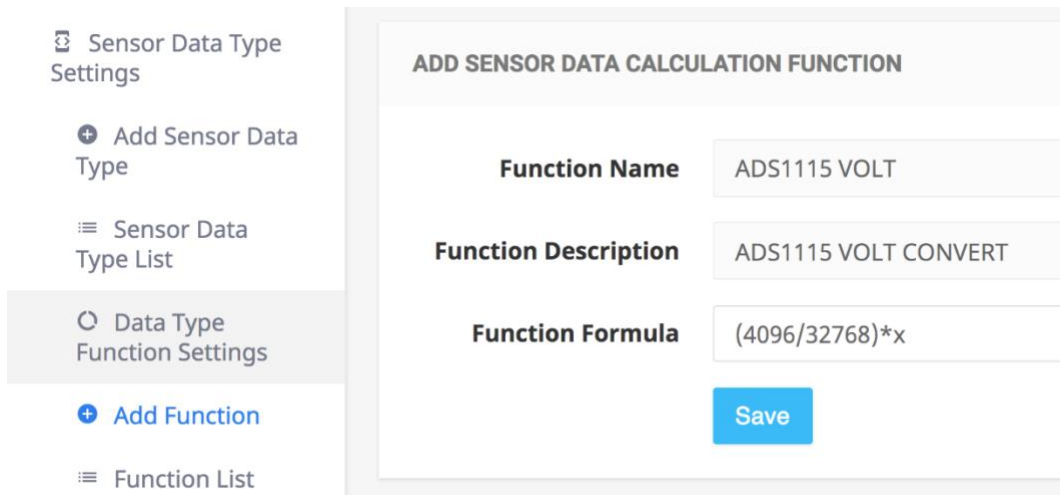


Figure 2.58: Sample function entry to the add function page.

The analog values to be received from this IC will be 2^{16} , but since the ADS1115 IC has negative receiving capability, this value will be between -32768 and 32767. Since the gain value is set to 1 during analog receiving in the Python script, the voltage value received will vary from -4096 to +4096. The formulas we need to perform to convert the data from the sensor to voltage are $Voltage = \frac{4096}{32768} * x$ if the "x" is treated as raw data coming from the python script. This equation should be entered into the system as shown in Figure 2.58.

2.8.7 Function list page

In this page, functional equations are listed as can be seen in Figure 2.59. Also, from this page, this data can be exported to PDF, EXCEL, CSV formats.

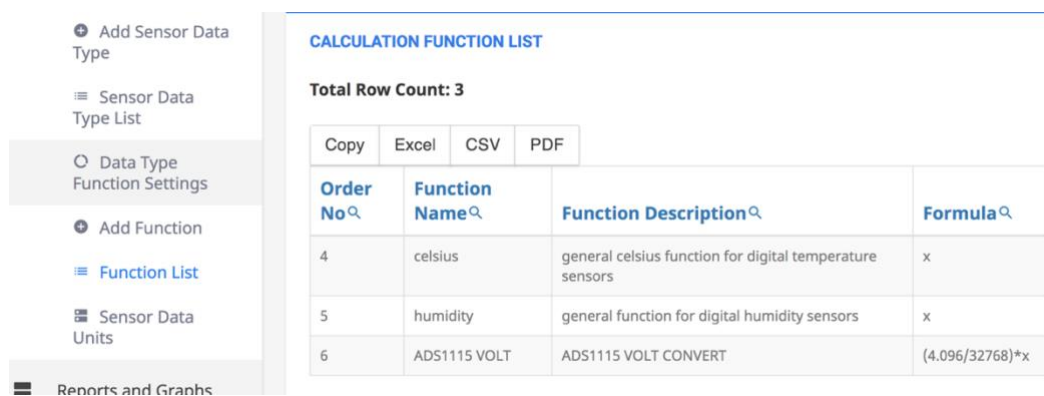


Figure 2.59: Function List page screenshot.

2.8.8 Add sensor data type page

In this page, the unification of the previously defined function and also the unit type definitions for a sensor type takes place. In the "Data Type Name" section, the data type name to be created is entered. In the "Sensor Type" section, the type of sensor to be used is entered. The unit that is added to the page shown in Figure 2.56 in the "Sensor Data Unit Type" section is selected. In the "Has Calculation Function" option, a "Yes" or "No" response is given in response to whether this data type should use a function. In the example shown in Figure 2.60, the function created previously Figure 2.58 is selected and the "Save" button is pressed. In the example shown in Figure 2.60, a data type designation is done which uses the function "ADS1115 VOLT" with the unit "volts" and can only work for analog sensors.

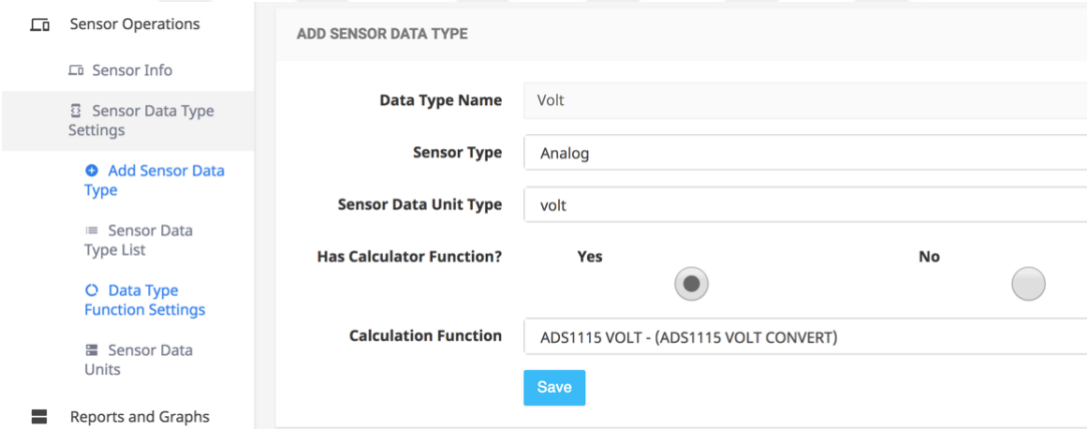


Figure 2.60: Add sensor data type page screenshot.

2.8.9 Sensor data type list page

In this page, the function definitions defined to the system as in the example that can be seen in Figure 2.60, in the "Add Sensor Data Type" page (Figure 2.61) is listed. From this page the user can also export this data to PDF, EXCEL and CSV file formats.

Sensor Operations

- Sensor Info
- Sensor Data Type Settings
 - Add Sensor Data Type
 - Sensor Data Type List**
 - Data Type Function Settings
 - Sensor Data Units

SENSOR DATA TYPE LIST

Total Row Count: 3

Copy Excel CSV PDF

Order No	Data Type Name	Sensor Type	Unit Name	Calculation Function	Function Name
6	Volt	Analog	volt	Var	ADS1115 VOLT - (ADS1115 VOLT CONVERT)
4	celsius	Digital	celsius	Var	celsius - (general celsius function for digital temperature sensors)
5	humidity	Digital	%	Var	humidity - (general function for digital humidity sensors)

Figure 2.61: Sensor data type list page screenshot.

2.8.10 Undefined sensor list page

This page lists the sensors that have not been added to the system (Figure 2.62) and this list can be downloaded to the computer in PDF, CSV or Excel formats.

If the sensor sent to the system via the Python script is not attached to the system, the name of the sensor is saved here. If the sensor that is needed to be added is selected by clicking on the button under the “Actions” row, the system will automatically route the user to the undefined sensor addition page as shown in Figure 2.63 to allow the selected sensor to be added to the system.

Mainpage

- Sensor Operations
 - Sensor Info
 - Add Sensor
 - Sensor List
 - Undefined Sensor List**
 - Sensor Data Type Settings
 - Reports and Graphs
 - Device Pin Settings
 - Admin Settings

UNDEFINED SENSOR LIST

Total Row Count: 9

Copy Excel CSV PDF

Order No	Sensor Number	Actions
11	ADS1115-A0	➔
12	10-000802189d4c	➔
13	ADS1115-A1	➔
14	sht11-H	➔
15	ADS1115-A2	➔
16	10-00080218867d	➔
17	sht11-T	➔
18	ADS1115-A3	➔
19	28-000006316ff2	➔

Figure 2.62: Undefined sensor list page screenshot.

Figure 2.63: Undefined sensor add page screenshot.

The "Sensor Number / ID" field on the undefined sensor addition page automatically pops up as filled according to the sensor selected in Figure 2.62. For example, if the "ADS1115-A0" sensor is selected, the sensor type is selected as "Analog" because it carries Analog data. Once this selection is made, the "Sensor Data Type" that was previously created in Figure 2.60 is selected for receiving the voltage from the ADS1115. Next, "Sensor Name" field needs to be filled. It is specified as "AN0" in Figure 2.63. In the next two sections, it is stated that the value from this sensor being lower than "3.2" is considered normal which means if the sensor value exceeds "3.2", an alarm will be shown on the main screen.

2.8.11 Add sensor page

A sensor which cannot be displayed on the "Undefined Sensor List" page since it is not yet connected to the system, can manually be added in the screen that is shown in Figure 2.64. While this area is being filled, the sensor ID of the sensor which is going to be installed later must be written correctly in this area. Otherwise, when the sensor is installed, the sensor is displayed on the "Undefined Sensor List" page and no data is saved in the system. The process in this area is identical to the process which will be carried out in Figure 2.63. The only difference between the two pages is that the "Sensor Number / ID" section is not auto-filled when the manual addition is in progress.

Figure 2.64: Add sensor page screenshot.

2.8.12 Sensor list page

In the example shown in Figure 2.65, the sensors registered in the system are listed together with all their basic definitions. The value of the sensors that appear in this list is saved into the Database. The desired sensor can be deleted under the Actions tab. If a sensor is deleted, the sensor data for that sensor is also deleted. From this page, the user can also export this data to PDF, EXCEL or CSV file formats.

Order No	Sensor Name	Sensor Type	Data Type Name	Unit Name	Actions
5	PROBE1	Digital	celsius	celsius	[Action]
7	PROBE2	Digital	celsius	celsius	[Action]
8	PROBE3	Digital	celsius	celsius	[Action]
9	TEMP-SHT	Digital	celsius	celsius	[Action]
6	Humidity	Digital	humidity	%	[Action]
10	AN0	Analog	Volt	volt	[Action]
11	AN1	Analog	Volt	volt	[Action]
12	AN2	Analog	Volt	volt	[Action]
13	AN3	Analog	Volt	volt	[Action]

Figure 2.65: Sensor list page screenshot.

3. RESULT

3.1 Checking the System Usage

Figure 3.1 shows the system resources that the system uses while running in full function. The CPU load is less than 10% because of the FFmpeg hardware accelerated video streaming. In Figure 3.2 “/var/www/nginx/html/hls/” folder while performing the HTTP live stream. This folder does not exceed 65 Mb during video broadcast, even though there is a reserved space of 100Mb on the ram.

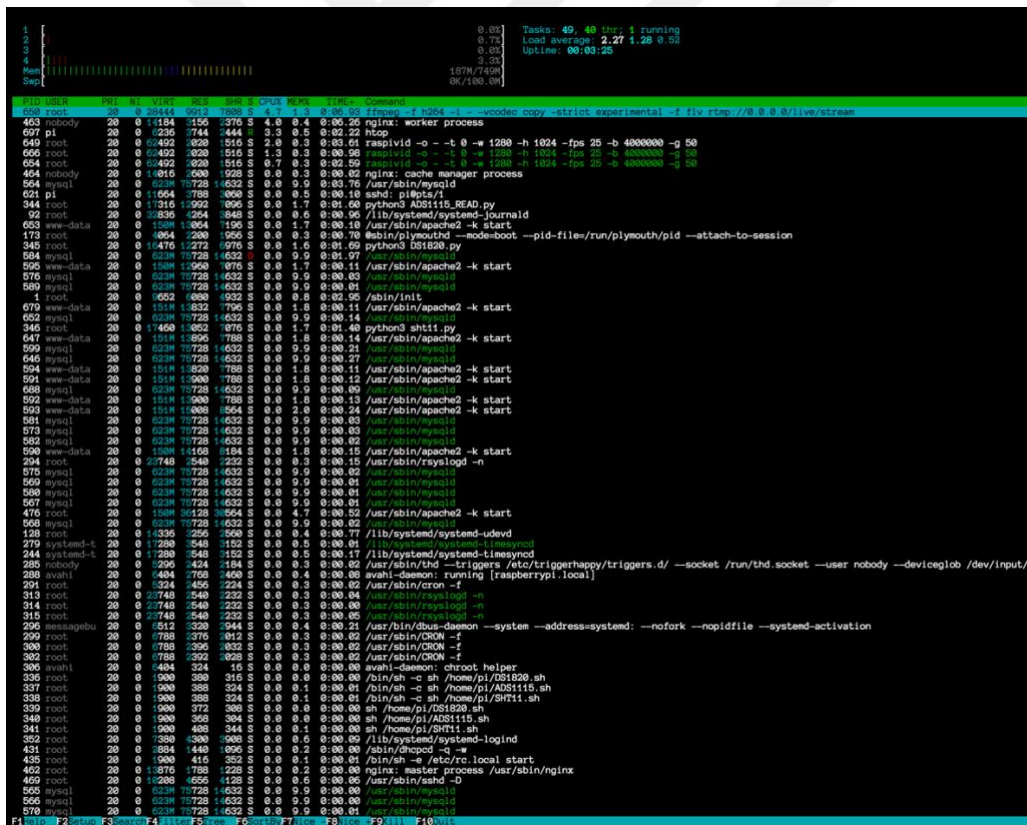


Figure 3.1: “htop” screenshot.

Although Apache and Nginx servers work at the same time, the system does not slow down. The only place where the system is slow enough to be noticed by the user is the page where the sensor data is listed and shown graphically as seen before in

3.2 Checking the Functionality of the "Device Pin Options" Field

When the user enters the system, the sensor data can be seen on the Main page. Immediately on the lower side, the user is able to see live images from the camera. In the right-hand area, the user can perform the triggering operations for the pins on the Raspberry Pi. By using "gpio -g readall" command through the terminal, the user can check whether the trigerring was successful or not by looking at the pins which were defined on the system before, through the BCM row. For example, when the "gpio -g readall" command is entered from the terminal after the situation that is shown in Figure 3.5 takes place, the output that the system will show will be something similar to Figure 3.6. From the screen that comes up, when the pin numbers in Figure 3.5 are found under the BCM column and the corresponding pin states are checked under the "V" column it will be seen that the pin states are set to "1".

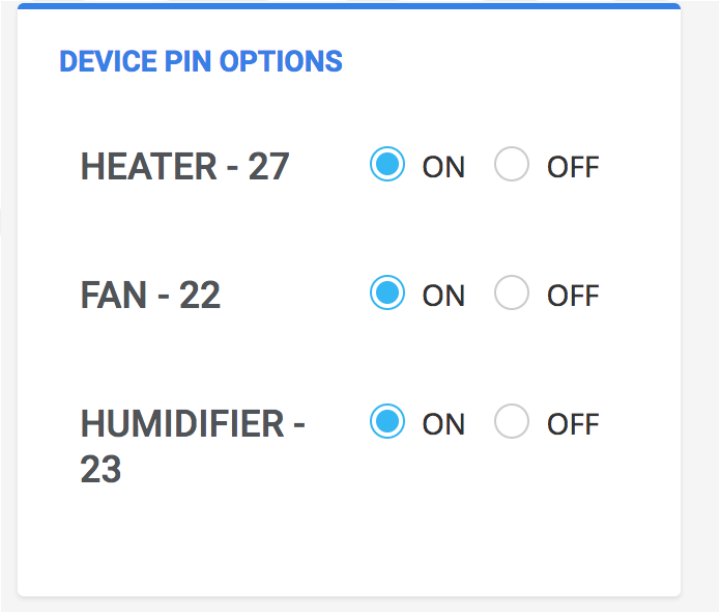


Figure 3.5: Pin triggering sample.


```
pi@raspberrypi:~$ gpio -g readall
```

Pi 2										
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM
		3.3v			1	2		5v		
2	8	SDA.1	ALT0	1	3	4		5v		
3	9	SCL.1	ALT0	1	5	6		0v		
4	7	GPIO.7	IN	1	7	8	1	ALT0	TxD	15
		0v			9	10	1	ALT0	RxD	16
17	0	GPIO.0	IN	0	11	12	1	IN	GPIO.1	1
27	2	GPIO.2	OUT	1	13	14		0v		18
22	3	GPIO.3	OUT	1	15	16	1	OUT	GPIO.4	4
		3.3v			17	18	0	IN	GPIO.5	5
10	12	MOSI	ALT0	0	19	20		0v		24
9	13	MISO	ALT0	0	21	22	0	IN	GPIO.6	6
11	14	SCLK	ALT0	0	23	24	1	OUT	CE0	10
		0v			25	26	1	OUT	CE1	11
0	30	SDA.0	IN	1	27	28	1	IN	SCL.0	31
5	21	GPIO.21	IN	1	29	30		0v		1
6	22	GPIO.22	IN	1	31	32	0	IN	GPIO.26	26
13	23	GPIO.23	IN	0	33	34		0v		12
19	24	GPIO.24	IN	0	35	36	0	IN	GPIO.27	27
26	25	GPIO.25	IN	0	37	38	0	IN	GPIO.28	28
		0v			39	40	0	IN	GPIO.29	29
										21

Figure 3.6: Gpio pin states.

This control can also be done by entering the “gpio -g read <BCM_Pin_number>” and checking the pin state for its value for being either “1” or “0”. For the situation that is shown in Figure 3.5, the screen output would be similar to Figure 3.7

```
pi@raspberrypi:~$ gpio -g read 27
1
pi@raspberrypi:~$ gpio -g read 22
1
pi@raspberrypi:~$ gpio -g read 23
1
```

Figure 3.7: “gpio -g read” command pin states.

This situation, shown in Figure 3.6 and also appears physically when looking at the status of the 3 LEDs placed on the circuit on the breadboard (Figure 3.8).

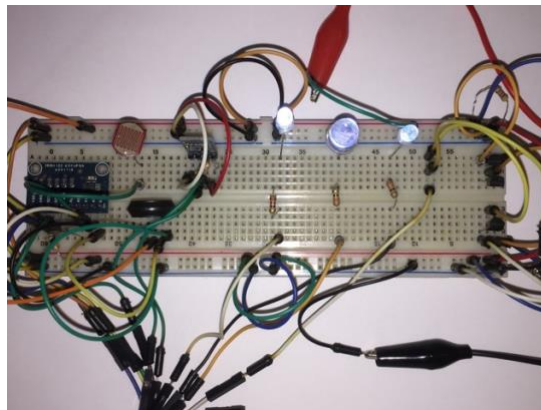


Figure 3.8: The circuit which is set-up on the breadboard.

3.3 Controlling the Camera Live Stream

Live broadcast from the camera is tested by setting up the mechanism which can be seen in Figure 3.9. In this simple set-up, the picture of a baby, which is displayed on a tablet-pc is broadcast via the camera. After the part of the video is taken from the camera, it is posted on the web page with a delay that is between 6 to 10 seconds. This is because the video that is recorded through the camera is split into 6-second clips and saved inside “/var/www_nginx/html/hls/” folder and the broadcast on the webpage needs to follow the live video more than 6 seconds behind. Examples of files recorded in 6-second intervals can be seen in Figure 3.10.

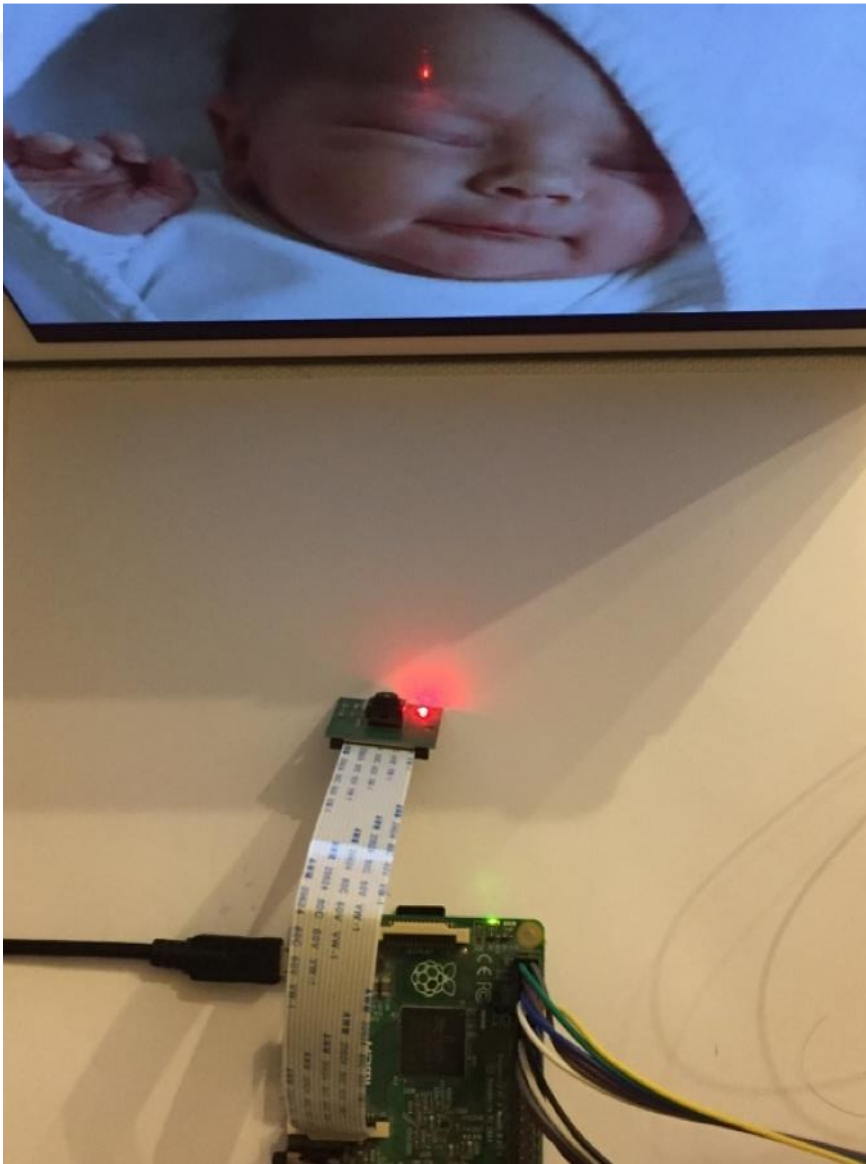


Figure 3.9: Live stream sample.

..			
stream.m3u8	256	m3u8-do...	14/08/2018 02:16:27
stream-4790.ts	143 820	MPEG-2 T...	14/08/2018 02:16:27
stream-4789.ts	3 081 884	MPEG-2 T...	14/08/2018 02:16:27
stream-4788.ts	3 080 756	MPEG-2 T...	14/08/2018 02:16:21
stream-4787.ts	3 086 208	MPEG-2 T...	14/08/2018 02:16:15
stream-4786.ts	3 077 372	MPEG-2 T...	14/08/2018 02:16:09
stream-4785.ts	3 082 824	MPEG-2 T...	14/08/2018 02:16:03
stream-4784.ts	3 082 636	MPEG-2 T...	14/08/2018 02:15:57
stream-4783.ts	3 082 072	MPEG-2 T...	14/08/2018 02:15:51
stream-4782.ts	3 082 072	MPEG-2 T...	14/08/2018 02:15:45
stream-4781.ts	3 084 140	MPEG-2 T...	14/08/2018 02:15:39
stream-4780.ts	3 087 336	MPEG-2 T...	14/08/2018 02:15:33

Figure 3.10: The ts files that are being used while hls is in progress.

3.4 Checking the Functionality of the Sensor Area

The test of analogue and digital sensors has been performed. In this section, 3 more ds18b20 sensors are added to the system. In addition, the ADS1115 is connected to AN0 pin of the IC with LDR through the voltage divider rule, to test the ADC IC. The AN1 is connected to the 3.3-volt supply from the Raspberry Pi, the AN2 is connected to the GND terminal of the system, the AN3 is connected to the analog pressure sensor. In this case, the values for AN1 and AN2 in the system should be 3.3 volts and 0 volts respectively. The values to be received from AN0 and AN3 would vary according to the ambient light and air pressure, respectively.

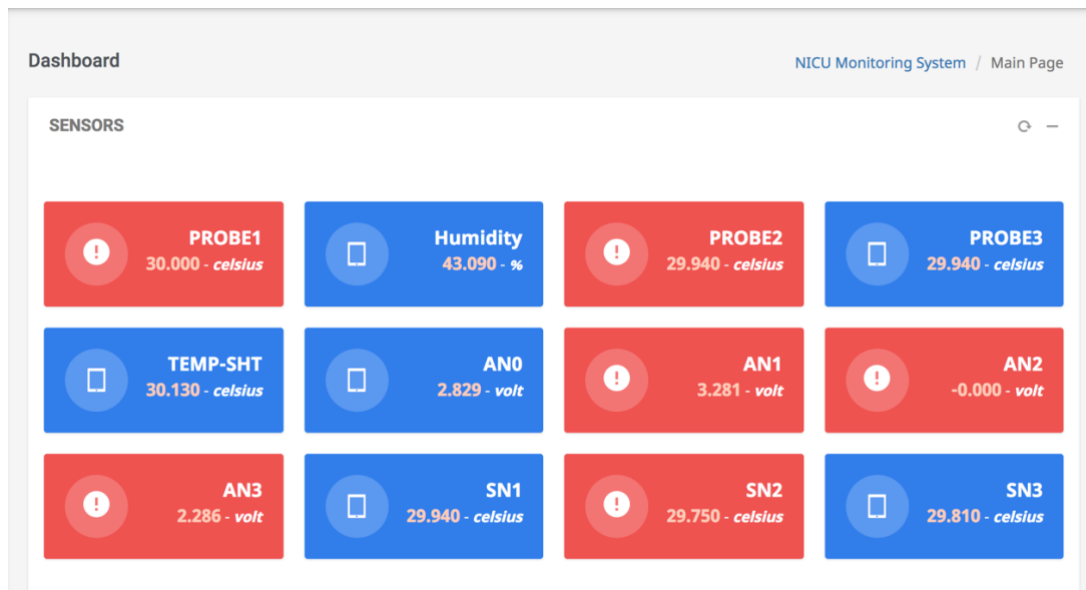


Figure 3.11: Sensor area.

The sensor field example which can be seen in Figure 3.11 is tested on a bright environment. Since the LDR sensor is connected to the GND side, AN0 pin has a value which is less than 3.3 volts. Since the AN1 probe is connected to 3.3 volts, it has the value 3.281 which is very close to it. The AN2 probe has a value of 0.000 because it is connected to the GND point. Since the AN3 probe is connected to the air pressure sensor, it receives the voltage value of the sensor according to the air pressure and shows 2.286. The LDR, which is connected to the AN0 probe, when not receiving light, has a value close to AN1, which is 3.283, as shown in Figure 3.12, which triggered the alarm condition because it did not meet the normal state condition previously defined for this sensor.

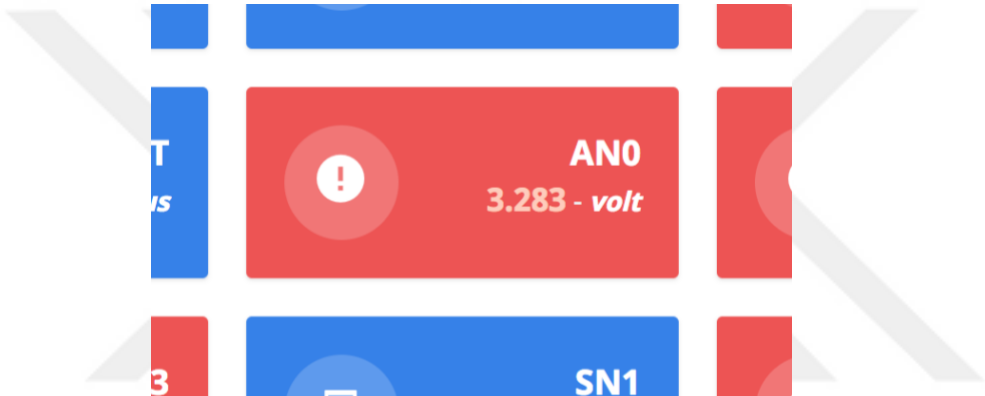


Figure 3.12: The 2nd state on AN0 probe.

The probes PROBE1, PROBE2, and PROBE3, which are shown in Figure 3.11, are the DS18B20 and DS18S20 sensors shown in Figure 2.62 before. The 3 DS18B20 sensors added for the purpose of system test are SN1, SN2, and SN3 sensors. Together with the recently added sensors, the system can receive data properly from all the defined sensors and record the received data in the database.

4. CONCLUSION AND DISCUSSION

The aim of this study is to develop a basic controller for incubators which would receive data from the sensors it is connected, would generate an alarm when needed, would control the devices that support the vitals of the baby, would allow the baby to be monitored remotely and, because of all these capabilities of the controller, would be critically important. Raspberry Pi is the main controller of the developed prototype.

Sensors which are used in this project are highly sensitive, have low noise and can be found easily and thus, can easily be put together by hospitals without an incubation system.

When the database table structure shown in Figure 3.4 is examined, it will be better understood how flexible the system is and how easy it is to develop.

The developed system has a structure that can be used separately as an environmental monitoring device.

While the system was being developed, there was a data corruption problem on the SD card due to the power failure which happened 3 times. In some of these failures, the Raspberry Pi could not boot while a database problem is occurred. Therefore, in case of a power failure, backing up the Raspberry Pi with a battery would be healthy not only for the data that the Raspberry Pi holds but also for the incubator.

While the system was being developed, 2 web servers, nginx, and apache, were used. Nginx normally provides PHP and SQL support, and when installed from Debian's package repository these features work fine. But after Nginx was compiled with the Rtmp server feature and installed on the system, PHP support could not be run. Therefore, the Apache server is installed on the system via the Debian package depository and used as the second and the PHP support server in the project.

The "crtmpserver" package from Debian's package repository can be installed and used on the system which would remove the need to install Nginx on the server. However, in the development phase of the project, no progress was made regarding "Crtmpserver".

It is thought that the developed system will be on an isolated network, which is safe from network attacks. For this reason, no work has been done on the network security side. Security vulnerabilities may be found and these may require extra work while patching.

When the chart for the sensors is requested over the web page, the waiting period would take a long time if the date range to be displayed is extended. Since the system tries to print all the date and time-based data on the screen, and because the chart is based on date-time, these date-time points overlap on each other and appear as a white line. In order to prevent this from happening and also to speed up the system, the sampling interval option has been added for graphical drawings. When graphics are requested from the system, this sampling interval must be specified manually. This sampling interval can be automatically determined by the system by developing the web page.

REFERENCES

- [1] Xia, Z., Brotman, R. M., Gajer, P., Abdo, Z., Schüette, U., Sam, M., . . . Forney, L. J. (2010). Recent Advances in Understanding the Microbiology of the Female Reproductive Tract and the Causes of Premature Birth. *Infectious Diseases in Obstetrics & Gynecology*, 2010, 1-10. doi:10.1155/2010/737425
- [2] Goldenberg, R. L., Hauth, J. C., & Andrews, W. W. J. N. E. j. o. m. (2000). Intrauterine infection and preterm delivery. *342*(20), 1500-1507.
- [3] Gonçalves, L. F., Chaiworapongsa, T., & Romero, R. (2002). Intrauterine infection and prematurity. *Mental Retardation and Developmental Disabilities Research Reviews*, 8(1), 3-13. doi:10.1002/mrdd.10008
- [4] Women's, N. C. C. f., & Health, C. s. (2011). Multiple pregnancy: the management of twin and triplet pregnancies in the antenatal period.
- [5] Seong, H. S., Jun, J. K., Ku, Y. H., Lee, S. E., Shim, S. S., Park, J. S., . . . Syn, H. C. (2006). OP11.12: The clinical implications of uterine myomas in preterm delivery; analysis based on antenatal ultrasonographic parameters. *28*(4), 486-486. doi:doi:10.1002/uog.3297
- [6] Heyborne, K., & Allshouse, A. A. (2016). Smoking and preterm birth: A novel effect of 17 alpha-hydroxyprogesterone caproate (17OHP-C). *Am. J. Obstet. Gynecol.*, 214(1), S220-S221.
- [7] WHO. (1997). *Thermal protection of the newborn: a practical guide*: World Health Organization: Geneva.
- [8] Hazan, J., Maag, U., & Chessex, P. (1991). Association between hypothermia and mortality rate of premature infants—Revisited. *American Journal of Obstetrics & Gynecology*, 164(1), 111-112. doi:10.1016/0002-9378(91)90638-8
- [9] Darmstadt, G. L., & Dinulos, J. G. (2000). NEONATAL SKIN CARE. *Pediatric Clinics*, 47(4), 757-782. doi:10.1016/S0031-3955(05)70239-X
- [10] Mittal, H., Mathew, L., & Gupta, A. J. I. J. E. T. E. E. (2015). Design and development of an infant incubator for controlling multiple parameters. *11*, 65-72.
- [11] Visscher, M. O., Adam, R., Brink, S., & Odio, M. J. C. i. d. (2015). Newborn infant skin: physiology, development, and care. *33*(3), 271-280.
- [12] Piermichele, P., & Simonetta, P. (2013). Apnea of prematurity. *Journal of Pediatric and Neonatal Individualized Medicine*, 2(2), e020213-e020213. doi:10.7363/020213
- [13] Agarwal, R., Chiswick, M. L., Rimmer, S., Taylor, G. M., McNally, R. J. Q., Alston, R. D., . . . Souza, S. W. (2002, 2002/03//). Antenatal steroids are associated with a reduction in the incidence of cerebral white matter lesions in very low birthweight infants. (Original Article). *Archives of Disease in Childhood Fetal and Neonatal Edition*, 86, F96+.
- [14] Meisels, S. J., Plunkett, J. W., Roloff, D. W., Pasick, P. L., & Stiefel, G. S. (1986). Growth and development of preterm infants with respiratory distress syndrome and bronchopulmonary dysplasia. *Pediatrics*, 77(3), 345-352.
- [15] Taesch, H., Ballard, R., & Gleason, C. (2005). *Avery's diseases of the newborn*. 8: Philadelphia: Elsevier.
- [16] White, R. J. E. o. M. D., & Instrumentation. (2006). Incubators, infant.

- [17] Mathanda, T. R., M. Bhat, R., Hegde, P., & Anand, S. (2015). Transepidermal Water Loss in Neonates: Baseline Values Using a Closed-Chamber System. *Pediatric Dermatology*, 33(1), 33-37. doi:10.1111/pde.12704
- [18] Brenda, H. M., Philbin, M. K., & Carl, B. (2000). Physiological Effects of Sound on the Newborn. *Journal of Perinatology*, 20(S1), S55. doi:10.1038/sj.jp.7200451
- [19] Vandenberg, K. A. (2007). Individualized developmental care for high risk newborns in the NICU: A practice guideline. *Early Human Development*, 83(7), 433-442. doi:10.1016/j.earlhumdev.2007.03.008
- [20] Lubbe, W., van der Walt, C., & Klopper, H. J. J. o. N. N. (2012). NICU environment—What should it be like? , 18(3), 90-93.
- [21] Aita, M., Johnston, C., Goulet, C., Oberlander, T. F., & Snider, L. (2013). Intervention Minimizing Preterm Infants' Exposure to NICU Light and Noise. *Clinical Nursing Research*, 22(3), 337-358. doi:10.1177/1054773812469223
- [22] Digital Humidity Sensor SHT1x. (2018).
- [23] Raspbian. (2018).
- [24] INSTALLING OPERATING SYSTEM IMAGES. (2018). Retrieved from <https://www.raspberrypi.org/documentation/installation/installing-images>
- [25] About FFmpeg. (2018). Retrieved from <https://www.ffmpeg.org/about.html>
- [26] FFmpeg Github. (2018). Retrieved from <https://github.com/FFmpeg/FFmpeg>
- [27] What is the Apache HTTP Server Project? (2018). Retrieved from https://httpd.apache.org/ABOUT_APACHE.html
- [28] About MariaDB. (2018). Retrieved from <https://mariadb.org/about/>
- [29] Adafruit Python ADS1x15. (2018). Retrieved from https://github.com/adafruit/Adafruit_Python_ADS1x15

CURRICULUM VITAE



Name Surname: Mehmet Hakan SELEK

Place and Date of Birth: İzmir / 05.07.1990

E-Mail: haselek@gmail.com

Education: 2014, Gediz University, Faculty of Engineering, Department of Electronic and Electrical Engineering

List of Publications:

Isler, Y., and Selek, M.H., Design of Incubator Control System with Online Video Streaming using Raspberry PI, International Conference of Applied Sciences, Engineering and Mathematics (IBU-ICASEM 2017), May 5-7, Struga-Ohrid/Macedonia, Book of Abstracts (ISBN: 978-608-65137-5-7), 89, 2017.