MODULAR “PLUG&PLAY” CONVEYOR SYSTEM

A Digital Engineering Project submitted by:

Akshatha Penagonde Sreenivasa Murthy, Mtr. no.: 217955

Jaydeepkumar G. Gondaliya, Mtr. no.: 214471

Mohammed Saif Sheikh, Mtr. no.: 213271

Moosa Shahid Mohammed, Mtr. no.: 218162

Pooja Ajit Kumar, Mtr. no.: 217944

Tina Ancelin Joseph Martin, Mtr. no.: 217945

Supervisors:

Dr.-Ing. Tobias Reggelin

M.Sc. Sebastian Lang

January 2018

MODULAR “PLUG&PLAY” CONVEYOR SYSTEM

Major: Digital Engineering

Abstract

With the wide use of embedded devices and improvement of the intelligence of the devices, the scale of conveyor systems in factories are becoming larger. These conveyor systems usually contain a lot of crossing or branching nodes, which leads to more complicated control. In the current practice, it is a more usually a central brain which controls the communication between the nodes in an entire system. Most of the industries use a “Centralized System”. In this project, we are developing a “Decentralized System” where in each component at every node will be controlled by its own brain. The goal is to develop a conveyor belt which will play a vital role in small scale as well as large scale industries for distributing the goods to a specific place, consequently reducing the cost of labor and time as well. A communication model with the help of Arduinos for the conveyor belts is designed. The Arduinos communicate with each other, this way the data is transmitted and received from the neighboring Arduino. Along with this, we are also realizing few arithmetic operations, polling concept and presenting the data over web interface. All the above concepts will be explained in detail in coming pages.

Keywords: Arduino, Conveyor Systems, Decentralized System

Table of Contents

Table of Figures

List of Tables

Chapter I

1. Introduction

Many industrial sectors, such as manufacturing, automotive, and material handling, are increasingly moving towards adopting modular conveyor systems in their processes since they offer significant flexibility in readily adapting to newer products and product lines, while making efficient use of available space, and these at a fraction of cost that otherwise would be incurred if an entirely new conveyor system is to be installed.

While using the modular conveyor systems in business sectors such as material handling used in industries like FedEx, UPS, and baggage handling in airport terminals, several questions need to be tackled. Few of them are mentioned in next few lines. What is the maximum sustainable rate of flow of goods in the system? Can handling of certain types of goods be prioritized over others? What is the impact of failures of certain sections of the conveyor system on the overall throughput and hence the monetary costs? How to plan inter material spacing on the conveyors such that the goods do not collide when they are switched through transfer elements (called turnarounds)?

To answer few of the above questions, we have developed a “Decentralized System” that involves the use of an Arduino on the conveyor belt. They use a variety of microprocessor and controllers, equipped with set of digital and analog input/output pins that may be interfaced to various expansion boards/circuits. The boards feature serial communication interfaces. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++.

The communication model depends on the working of an Arduino which plays an important role in the entire system. The Arduino is responsible for the communication (transfer of data). To the development of the decentralized system, the Arduinos are placed on the conveyor belt. Every time the conveyor units are sent to the next position, data transmission will be initiated to the neighboring Arduino and this way the communication is taken place. This system can carry conveyor units (for example, small load bearers) without any centralized infrastructure.

Chapter II

2. Background Work and Research

There has been a long history in the area of conveyor system research and application [1]. Traditional unit-load converyor systems generally consists of several modular roller, belt or chain converyors, which is characterized by small scale, single function, and single running direction, all of this results in the lower efficiency of the conveyor system. Although there are some crossing and branching nodes in the sorting systems used in some large postal, logistic factories or distribution centers [2], the running direction is still single, thus the control algorithm and the program is relatively simple, and the conventional centralized algorithm may satisfies entrire system well.

Research on the converyor system can be divided into hardware design , manufacturing, process control as well as specialized applications. In 1933, CEMA(Conveyor Equipment Manufacturers Association) [3] was established to draw up the reference standard for the design of roller, belt, chain and spiral conveyor, it also set the standard on the safety, emergency stop and noice control during the operation of the conveyor system. The control of converyor system is always the focus of discussion. Design and planning of an airport terminal automatic transmission system is described in [4], which discusses the use of different converyor devices, but they are all the simple ferry way with single direcction control. The design, routing, scheduling problem of ins or outs of the storage in the AS/RS(Automated Storage and Retrieval System) is discussed in [5] which contains only several intersections, which is simple analysis and application of the conveyor system. In [6], the authors describe the optimal routing and scheduling in the conveyor network based on ant algorithm, which is a method of centralized control. A decentralized control system is described in [7] and there are also many literatures discuss the analysis and improvement of the dedicated sortation system [8].

In the application, literature [9] analyses the operation performance of picking in automated warehouse delivery system with multiple IO, and the improved model is given. The application of conveyors in modern semiconductors wafer FABs is analysed comprehensively in [10]. A linear time approximation algorithm with minimum transfer time used in scheduling problem of flowshop is presented in [11]. In [12], author describes the terms and design of the modular conveyor system, a configurable simulation module.

Currently the scale of conveyor systems in some common factories is relatively small with a single transfer direction and the control algorithm used in these systems is commonly simple and conventional. For example, the shortest path and time for transferring is only considered in the case that only one unit-load is transferred, but the complex conveyor system with multi-branching-nodes is seldom considered, and the previous studies almost all focussed on centralized control. Comparitively the research on the conflict-free and optimal path in the case of multi unit transfer simultaneously is rare.

Most of the systems which are in practice has one central brain which will act as the control system for the whole system and will bear the information of every active node in the system.

The main intention of this project is to develop a “Decentralized System” where in each component at every node will be controlled by its own brain. One very good advantage of developing this system is that, any component which fails in the system can be easily replaced without disturbing the functioning of whole system within no time and by bearing very less effort.

We began research by searching for an open source computer hardware which helps in building digital devices and interactive objects that can sense and control objects in the physical world. We came across one such open source computer hardware, an “Arduino” which was feasible with our requirements. Over the years, Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments.

The focus of the project was to develop a communication model and after a detailed study of the components to be used, we then decided the major components to be used along with Arduino to develop a fully pledged model. This project required the understanding of all the electronic components and also the software used for an Arduino to perform tasks. The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application and the language is merely a set of C/C++ functions that can be called from your code. It also supplies a software library from the Wiring project, which provides many common input and output procedures. We decided to program our code in C since it is easily understandable and also explainable.

For the Arduino to transmit the data or to swap the data, they must share a common communication protocol. The whole purpose of a serial interface is to provide a single path for data transmission wirelessly or over a cable. Parallel buses are still used in some applications. There are dozens of serial data interfaces which are developed for specific applications. A few have become universal, such as I2C, CAN, LIN, SPI, Flex, MOST, and I2S.

Initially we started with using I2C protocol for establishing communication between aurdinos using Master slave principle where in the each of the Arduino’s can either act like a master or a slave. However due to limitations such as possiblity of address conflicts and slower speeds, we then switched over to RS485 serial communication. RS485 Serial Communication protocol is capable of transmitting data to maximum distance of 4000m or 4km (13). Since RS485 is asychronous, no clock signal is needed. Hence it can be used in noisy enviroments such as industries. Differences in use of RS485 over I2C communication:

(a) I2C requires less hardware since there are direct connections to every arduinos, however this leads to decrease in speed rate and also fewer distances (about 100m). RS485 can go for much longer distances (1.2 km).
(b) I2C supports Wire library which enables Arduinos to have an address of their own. Wire.beginTransmission(address). However, RS485 does not support Wire library, so the addressing needs to be done manually.
(c) I2C is synchronous whereas RS485 is asynchronous (no clock signal is used).

(d) I2C communication stops working when there are large number of arduinos connected with each other. Since every arduino has its own RS485 connected to it, it supports multiple receivers.
(e) I2C speed varies from 100kbit/s to 400kbit/s whereas RS485 can go upto 10 Mbit/s and maximun data rate is 30 Mbps.

2.1 SoftwareSerial library

The SoftwareSerial library has been used in our project to allow serial communication on other digital pins of the Arduino, using software to replicate the functionality (hence the name "SoftwareSerial"). It is possible to have multiple software serial ports with speeds up to 115200 bps. For Arduino Mega 2560, these communicate on pins 10 and 11 respectively. 10 being the receiving pin and 11 being the transmitting pin. This is integrated along with “RS485_protocol.h” library. This library was created specifically for ModBus RS485 by Nick Gammon (see references for link). The Software Serial library in Mega 2560 allows to send for multiple digital writes and also to receive for multiple digital reads.

The complete sketch for Arduino Mega 2560’s with RS485 is given below;

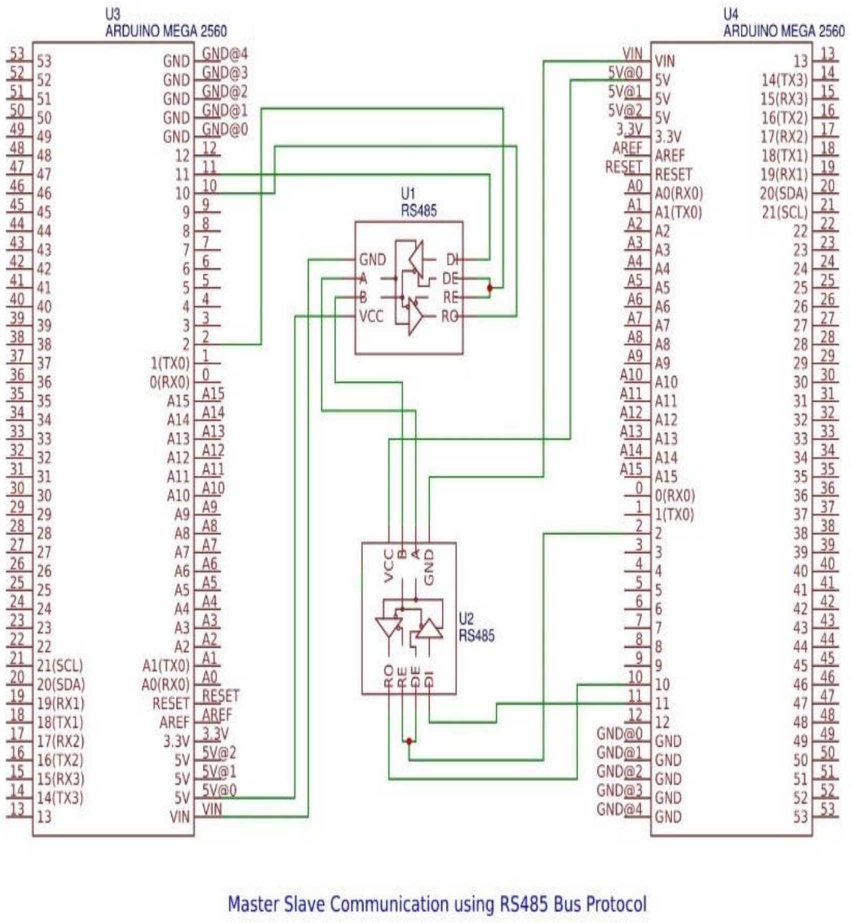


Figure 1 Sketch of Arduinos with RS485

Connection:

Arduino pin 10 (receive pin) is connected to Receive Output (RO) of RS485.

Arduino pin 11 (transmit pin) is connected to Digital Input (DI) of RS485.

Arduino 5V is connected to VCC of RS485 – to power up the RS485 through Arduino

Arduino VIN is connected to GND (Ground) of RS485 – to keep common grounds as to supply power

Arduino pin 2 (output pin) is connected to DE(Driver Enabled)/RE(Receiver Enabled) (jumpered together)

Arduinos are connected to each other using A’s and B’s of their respective RS485’s which are connected together.

With all the specifications in hand we started to develop the communication model and the results are discussed accordingly in this paper.

Chapter III

3. Objectives:

The objective is to implement a Master Slave Communication model which includes following functionalities.

1. **Communication Bridge between the Arduinos:** As per the requirement, the communication protocol used here is RS485. In that accordance, initially Master will request the Slave for a chunk of data, and the slave should respond accordingly. This explains the basic ideology behind the communication of Arduinos. The key aspect of this objective helps in achieving the following tasks.
   i) Addressing the Arduino: Each Arduino will be addressed by its unique address in the bus. The master is assigned with address 0 and rest of the slaves is addressed in the increasing order of int. The Arduinos can be referred using this address at any part of the code.
   ii) Transferring the Distance Matrix: Distance matrix helps in determining the distance between the existing slave Arduinos once the master raises the request for the same. That is, when master requests a specific slave for its distance, the requested slave sends its distance value with respect to all the other slaves from the matrix table to the master.
   iii) Transferring the Module Status Matrix: The position of the respective slave Arduino can be determined by keeping the conveyor belt position as the reference. The left side of the Conveyor belt would be ‘-1’ and the right side of the belt is ‘+1’. In this context, the master Arduino requests a specific slave Arduino for its module status with respect to all the other slaves, the requested slave sends its status from the module matrix table to the master. This way the Module Status Matrix of all the Slave Arduinos will be transferred to the Master respectively.
   iv) Arithmetic Operations: Arithmetic operations such as addition, subtraction, division, multiplication, minimum/maximum of two numbers are carried out here. Master will send an arithmetic expression to a specific slave, and the respective slave will perform a requested arithmetic operation. Once that is achieved, the slave will send the obtained result to the master.
   v) Polling: The master Arduino repeatedly polls each slave to send their values. The master requests cyclically to all slaves present in the neighborhood. The slaves then respond to the request with their values with the request key. In this manner, there will not be any collision of messages.

2. **Web Interface:** It describes to store/retrieve the values onto/from MySQL table while Arduino program is executing. The values will be inserted/updated based on the primary key values. To perform web interface, we make use of Raspberry Pi.

Chapter IV

4.Transferring Distance Matrix

As discussed earlier, transferring the distance values from slaves to the master is the main objective. Initially, Master will address each slave with a slave ID. That is, the master will request a specific slave to send its distance matrix by sending the slave ID to all the slaves. The slave’s ID which is matched with the requested ID from the master will reply by sending the distance values from the matrix table. For example, considering the below table with five slaves, when master requests for the distance matrix values with the slave ID 1, the ID will be sent to all the associated slaves. Once the ID matches with the slave, in this case which is slave 1 responds to the master by sending distance values for all possible combinations with other four slaves. That is, distances with respect to 1-1, 1-2, 1-3, 1-4 and 1-5 will be sent.

DISTANCE MATRIX TABLE

ARDUINO DISTANCES(SLAVES)	1	2	3	4	5
1	0	5	8	6	3
2	5	0	4	7	10
3	8	4	0	3	6
4	6	7	4	0	2
5	3	10	6	2	0

Table 1 Distance Matrix Table

Code Snippet for transferring distance matrix values between master and slave is given below.

MASTER CODE

```
#include "RS485_protocol.h"
#include <SoftwareSerial.h>
const byte ENABLE_PIN = 2;
SoftwareSerial rs485 (10, 11); // receive pin, transmit pin

char input[5];
int charsRead = 0;
boolean received = false;
char res;
boolean sendnow = false;



void setup()
{

  Serial.begin(9600);
  rs485.begin (28800);
  pinMode (ENABLE_PIN, OUTPUT);

}

void loop()
{

  Serial.flush();
  if (Serial.available() == 0)
  {

  }
  delay(200);
  while (Serial.available() > 0)
  {

    charsRead = Serial.readBytesUntil('\n', input, sizeof(input) - 1);
    sendnow = true;
  }
  if (sendnow)
  {
    digitalWrite (ENABLE_PIN, HIGH);  // enable sending
    rs485.write(input);
    Serial.println(input);
    //delayMicroseconds (660);
    digitalWrite (ENABLE_PIN, LOW);
    sendnow = false;
  }

  delay(1000);
  digitalWrite (ENABLE_PIN, LOW);

  while (rs485.available() && !sendnow)
  {
    Serial.println("Receiving Distance Matrix");
    int dist;

    for (int i = 0; i < 5; i++)
    {
      dist = rs485.read();
      digitalWrite (ENABLE_PIN, HIGH);

      Serial.println(dist);
    }

  }
```

```
}
```

Table 2 Master code for Distance Matrix

SLAVE1 CODE

```
#include "RS485_protocol.h"
#include <SoftwareSerial.h>
const byte ENABLE_PIN = 2;
SoftwareSerial rs485 (10, 11);  // receive pin, transmit pin
boolean replay = false;
int myDistValues[5] = {0, 5, 8, 6, 3};
int i;

void setup()
{
  Serial.begin(9600);
  rs485.begin (28800);
  pinMode (ENABLE_PIN, OUTPUT); // register event
}
void loop()
{
  digitalWrite (ENABLE_PIN, LOW);
  while (rs485.available())
  {
    char c = rs485.read();
    delay(10);
    if (c == '1') //1st slave ID
    {


      Serial.println("Sending from slave:" );
      Serial.println(c);
      digitalWrite (ENABLE_PIN, HIGH);
      replay = true;
    }



    delay(600);



    if (replay)
    {
      digitalWrite (ENABLE_PIN, HIGH);//enable sending to the master
      Serial.println("Sending 1st slave Distance Matrix");
      for (i = 0; i < 5; i++) {
        rs485.write(myDistValues[i]);
        Serial.println(myDistValues[i]);
      }
      digitalWrite (ENABLE_PIN, LOW);
    }
    replay = false;
  }
}
```

Table 3 Slave 1 code for Distance Matrix

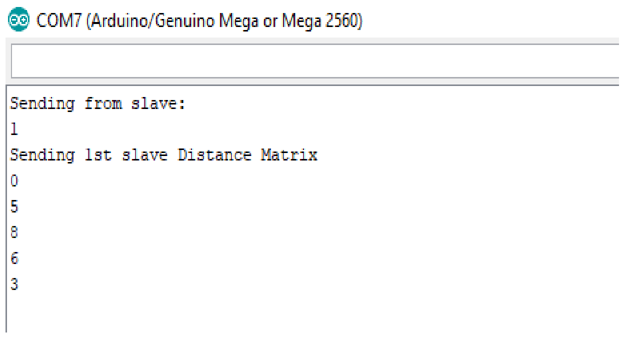
SLAVE2 CODE

```
#include "RS485_protocol.h"
#include <SoftwareSerial.h>
const byte ENABLE_PIN = 2;
SoftwareSerial rs485 (10, 11);  // receive pin, transmit pin
boolean replay = false;
int myDistValues[5] = {5, 0, 4, 7, 10};
int i;

void setup()
{
  Serial.begin(9600);
  rs485.begin (28800);
  pinMode (ENABLE_PIN, OUTPUT); // register event
}
void loop()
{
  digitalWrite (ENABLE_PIN, LOW);
  while (rs485.available())
  {
    char c = rs485.read();
    delay(10);
    if (c == '2') //2nd slave ID
    {


      Serial.println("Sending from slave:" );
      Serial.println(c);
      digitalWrite (ENABLE_PIN, HIGH);
      replay = true;
    }

    delay(600);

    if (replay)
    {
      digitalWrite (ENABLE_PIN, HIGH);//enable sending to the master
      Serial.println("Sending 2nd slave Distance Matrix");
      for (i = 0; i < 5; i++) {
        rs485.write(myDistValues[i]);
        Serial.println(myDistValues[i]);
      }
      digitalWrite (ENABLE_PIN, LOW);
    }
    replay = false;
  }
}
```

Table 4 Slave 2 code for Distance Matrix

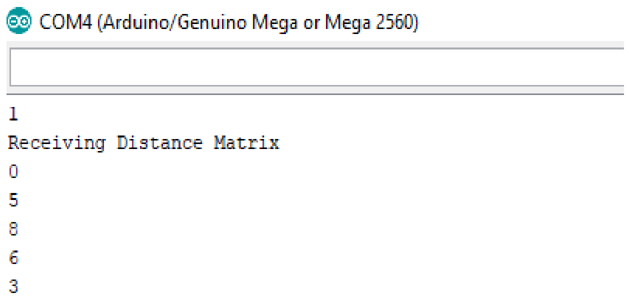
4.1 Execution:

1. Master will address each slave with a slave ID by sending a character value by sending the slave ID to all the slaves. For example, considering the below output window slave 1 is addressed. The master sends a character '1' as shown in the below output window.
2. Once the Master sends the slave ID, the slave's ID which is matched with the requested ID from the master will reply by sending the distance values from the matrix table. In our case the slave 1 will print its distance values from the matrix table and will respond back to the master as shown in the output windows. The master will finally receive the distance matrix from the requested slave. The output on the serial monitor of the master and the slave is show below in the pictures.



```
Sending from slave:
1
Sending 1st slave Distance Matrix
0
5
8
6
3
```

Figure 2 Output for Slave 1 – Distance Matrix



```
1
Receiving Distance Matrix
0
5
8
6
3
```

Figure 3 Output for Master – Distance Matrix

3. Similarly, when the master address slave 2 by requesting the slave 2 to send its distance matrix values, slave 2 will respond back and there is no response from the other slaves

4.2 Sequence Diagram:

The below figure shows the sequence diagram for transferring the distance matrix table values between Master and Slaves. As discussed earlier, in the beginning Master sends the request to each slave to get their Distance back to the Master. Selection of each Slave is done by User by using Serial Monitor of Master. When User input is 1 then only Slave1 is sending its distance to the Master which will be displayed in Serial monitor of Master whereas, Slave2 also receives the request from Master but it will not send its distance to the Master during this time.

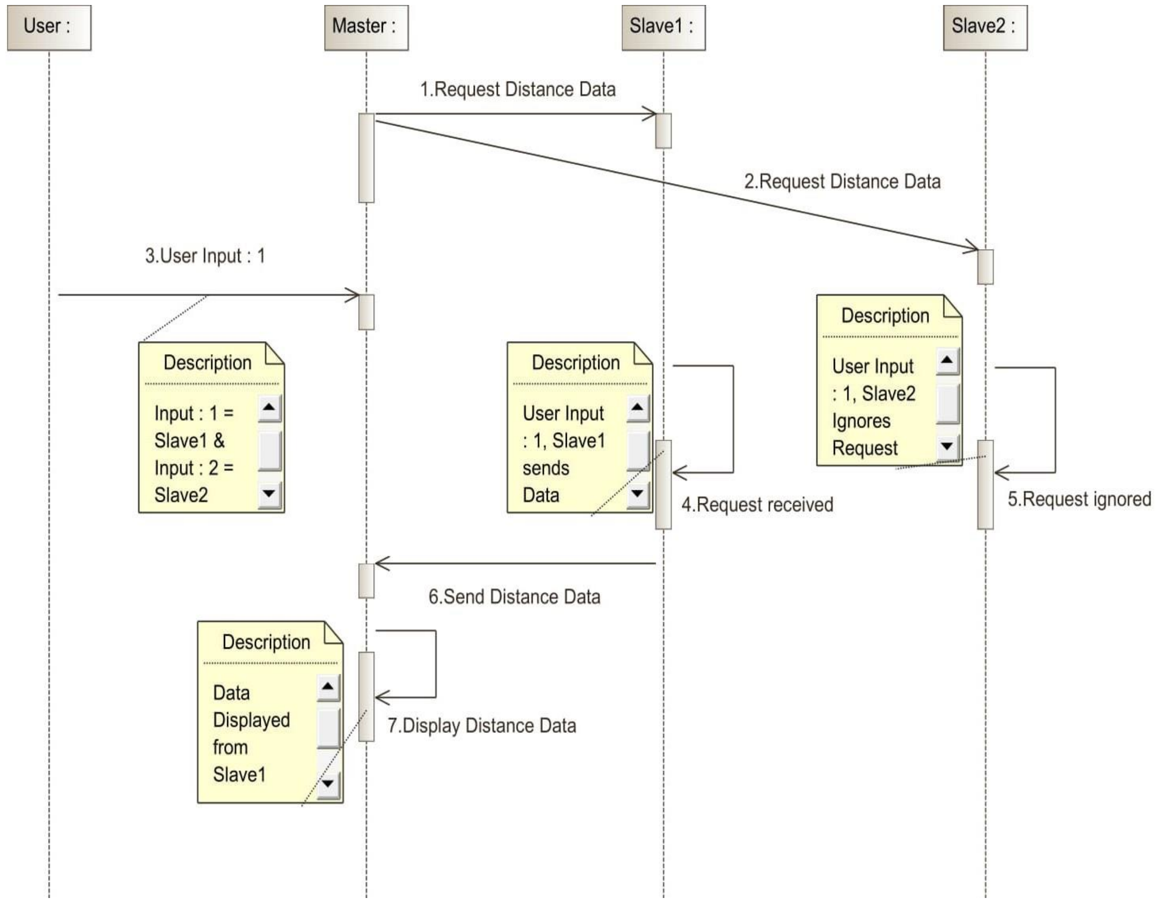


Figure 4 Sequence Diagram – Distance Matrix

Chapter V

5. Transferring Module Status Matrix

The module-status matrix has same size as that of distance matrix. Here the module-status of the Arduino changes with respect to the object’s movement in the conveyer belt. There are 3 possible states for each Arduino. The states are 0, 1 and -1. 0 indicates that the module has only one position state, 1 and -1 are respective first and second position states of the module. The module status matrix needs to be updated at every position of the movement of the object. This can be explained with a figure.

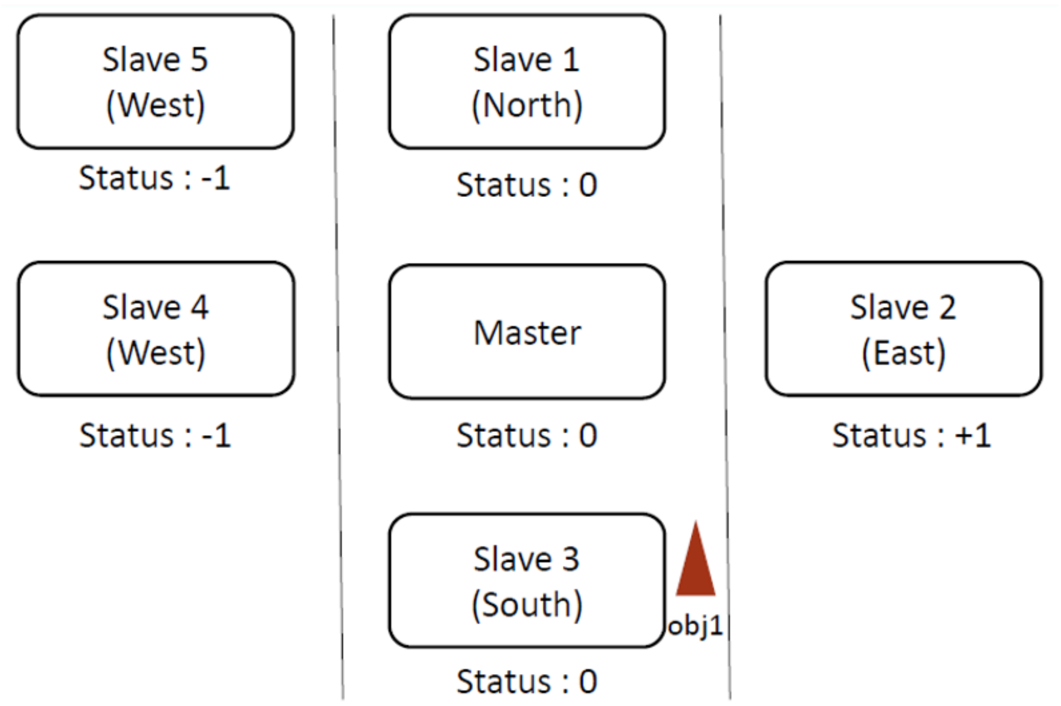


Figure 5 Modulo Status Matrix

The above given figure is of a conveyer belt with Arduinos at every end. There are one master present and remaining slaves. If object (obj1) is present at Slave 3 at initial point, the status of other Arduinos in that position will be kept 0 and to the right side of slave 3 i.e., slave 2’s status is +1 and to the left side of the slave 3 i.e., slave 4’s and slave 5’s status is -1.

Now, the distance will be calculated by the master and it chooses the best possible distance for obj1 to travel from slave 3 to other slaves. If slave 3 to slave 2 has the best possible distance among all the distances, then the obj1 travels to slave 2. From slave 2, the status changes again, i.e., slave 2, master and slave 4 will have its status: 0. To the right side, i.e., slave 5 and slave 1 will have status as +1 and on the left side, i.e., slave 3 will have its status -1. In this order, the communication continues and the module status matrix changes. Since at every position the module status matrix needs to be updated, the Arduinos will have pre-defined module status matrices. These matrices will be different from each Arduinos.

ARDUINO STATUS	1	2	3	4	5
1	0	1	0	-1	-1
2	1	0	-1	0	1
3	0	1	0	-1	-1
4	-1	0	1	0	-1
5	0	1	1	1	0

Table 5 Modulo Status Matrix Table

The communication between master and slave for transferring module status matrix works precisely as explained in the above section “Transferring Distance Matrix”.

Code Snippet for transferring module-status matrix values between master and slave is given below.

MASTER CODE

```
#include "RS485_protocol.h"
#include <SoftwareSerial.h>
const byte ENABLE_PIN = 2;
SoftwareSerial rs485 (10, 11); // receive pin, transmit pin

char input[5];
int charsRead = 0;
boolean received = false;
char res;
boolean sendnow = false;



void setup()
{

  Serial.begin(9600);
  rs485.begin (28800);
  pinMode (ENABLE_PIN, OUTPUT);

}

void loop()
{

  Serial.flush();
  if (Serial.available() == 0)
  {

  }
  delay(200);
  while (Serial.available() > 0)
  {

    charsRead = Serial.readBytesUntil('\n', input, sizeof(input) - 1);
    sendnow = true;
  }
  if (sendnow)
  {
    digitalWrite (ENABLE_PIN, HIGH);  // enable sending
    rs485.write(input);
    Serial.println(input);
    //delayMicroseconds (660);
    digitalWrite (ENABLE_PIN, LOW);
    sendnow = false;
  }

  delay(1000);
  digitalWrite (ENABLE_PIN, LOW);

  while (rs485.available() && !sendnow)
  {
    Serial.println("Receiving Module Status Matrix");
    int mod;
    for (int i = 0; i < 5; i++) {
      mod = rs485.read();
      digitalWrite (ENABLE_PIN, HIGH);
      Serial.println(mod);
    }
  }
}
```

Table 6 Master code for Modulo Status Matrix

SLAVE1 CODE

```
#include "RS485_protocol.h"
#include <SoftwareSerial.h>
const byte ENABLE_PIN = 2;
SoftwareSerial rs485 (10, 11);  // receive pin, transmit pin
boolean replay = false;
int myStatusValues[5] = {0, 1, 0, -1, -1};
int i;

void setup()
{
  Serial.begin(9600);
  rs485.begin (28800);
  pinMode (ENABLE_PIN, OUTPUT); // register event
}
void loop()
{
  digitalWrite (ENABLE_PIN, LOW);
  while (rs485.available())
  {
    char c = rs485.read();
    delay(10);
    if (c == '1') //1st slave ID
    {


      Serial.println("Sending from slave:" );
      Serial.println(c);
      digitalWrite (ENABLE_PIN, HIGH);
      replay = true;
    }



    delay(600);


    if (replay)
    {
      digitalWrite (ENABLE_PIN, HIGH);//enable sending to the master
      Serial.println("Sending 1st slave Module Status Matrix");
      for (i = 0; i < 5; i++) {
        rs485.write(myStatusValues[i]);
        Serial.println(myStatusValues[i]);
      }
      digitalWrite (ENABLE_PIN, LOW);
    }
    replay = false;
  }
}
```

Table 7 Slave 1 code for Modulo Status Matrix

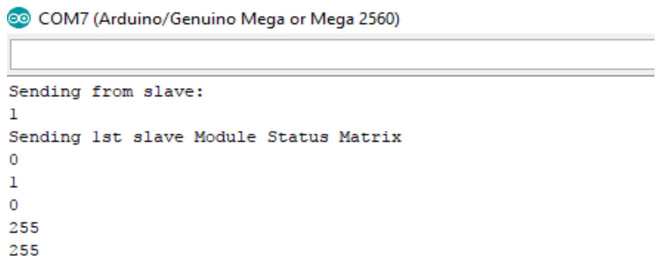
SLAVE2 CODE

```
#include "RS485_protocol.h"
#include <SoftwareSerial.h>
const byte ENABLE_PIN = 2;
SoftwareSerial rs485 (10, 11);  // receive pin, transmit pin
boolean replay = false;
int myStatusValues[5] = {1, 0, -1, 0, 1};
int i;

void setup()
{
  Serial.begin(9600);
  rs485.begin (28800);
  pinMode (ENABLE_PIN, OUTPUT); // register event
}
void loop()
{
  digitalWrite (ENABLE_PIN, LOW);
  while (rs485.available())
  {
    char c = rs485.read();
    delay(10);
    if (c == '1') //1st slave ID
    {


      Serial.println("Sending from slave:" );
      Serial.println(c);
      digitalWrite (ENABLE_PIN, HIGH);
      replay = true;
    }



    delay(600);


    if (replay)
    {
      digitalWrite (ENABLE_PIN, HIGH);//enable sending to the master
      Serial.println("Sending 2nd slave Module Status Matrix");
      for (i = 0; i < 5; i++) {
        rs485.write(myStatusValues[i]);
        Serial.println(myStatusValues[i]);
      }
      digitalWrite (ENABLE_PIN, LOW);
    }
    replay = false;
  }
}
```

Table 8 Slave 2 code for Modulo Status Matrix

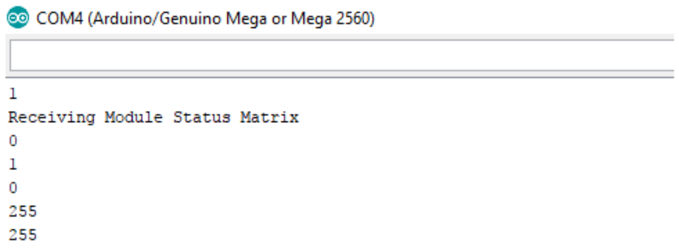
5.1 Execution:

1. Master will address each slave with a slave ID by sending a character value by sending the slave ID to all the slaves. For example, considering the below output window slave 1 is addressed. The master sends a character '1' as shown in the below output window.
2. Once the Master sends the slave ID, the slave's ID which is matched with the requested ID from the master will reply by sending the module status values from the matrix table. In our case the slave 1 will print its status values from the matrix table and will respond back to the master as shown in the below output window. The master will finally receive the module status matrix from the requested slave. The output on the serial monitor of the master is show below in the pictures.



```
COM7 (Arduino/Genuino Mega or Mega 2560)

Sending from slave:
1
Sending 1st slave Module Status Matrix
0
1
0
255
255
```

Figure 6 Output for Slave 1 Modulo Status Matrix



```
COM4 (Arduino/Genuino Mega or Mega 2560)

1
Receiving Module Status Matrix
0
1
0
255
255
```

Figure 7 Output for Master Modulo Status Matrix

3. Similarly, when the master address slave 2 by requesting the slave 2 to send its distance matrix values, slave 2 will respond back and there is no response from the other slaves. Note that the execution of the Module status matrix is similar to that of the distance matrix

5.2 Sequence Diagram:

The figure below shows the sequence diagram for transferring the Modulo Status matrix values between Master and Slaves. In the beginning, Master sends the request to each slave to get their Status back to the Master. Selection of each Slave is done by user by using Serial Monitor of Master. When User input is 1 then only Slave1 is sending its Status to the Master which will be displayed in Serial monitor of Master whereas, Slave2 also receives the request from Master but it will not send its Status to the Master during this time.

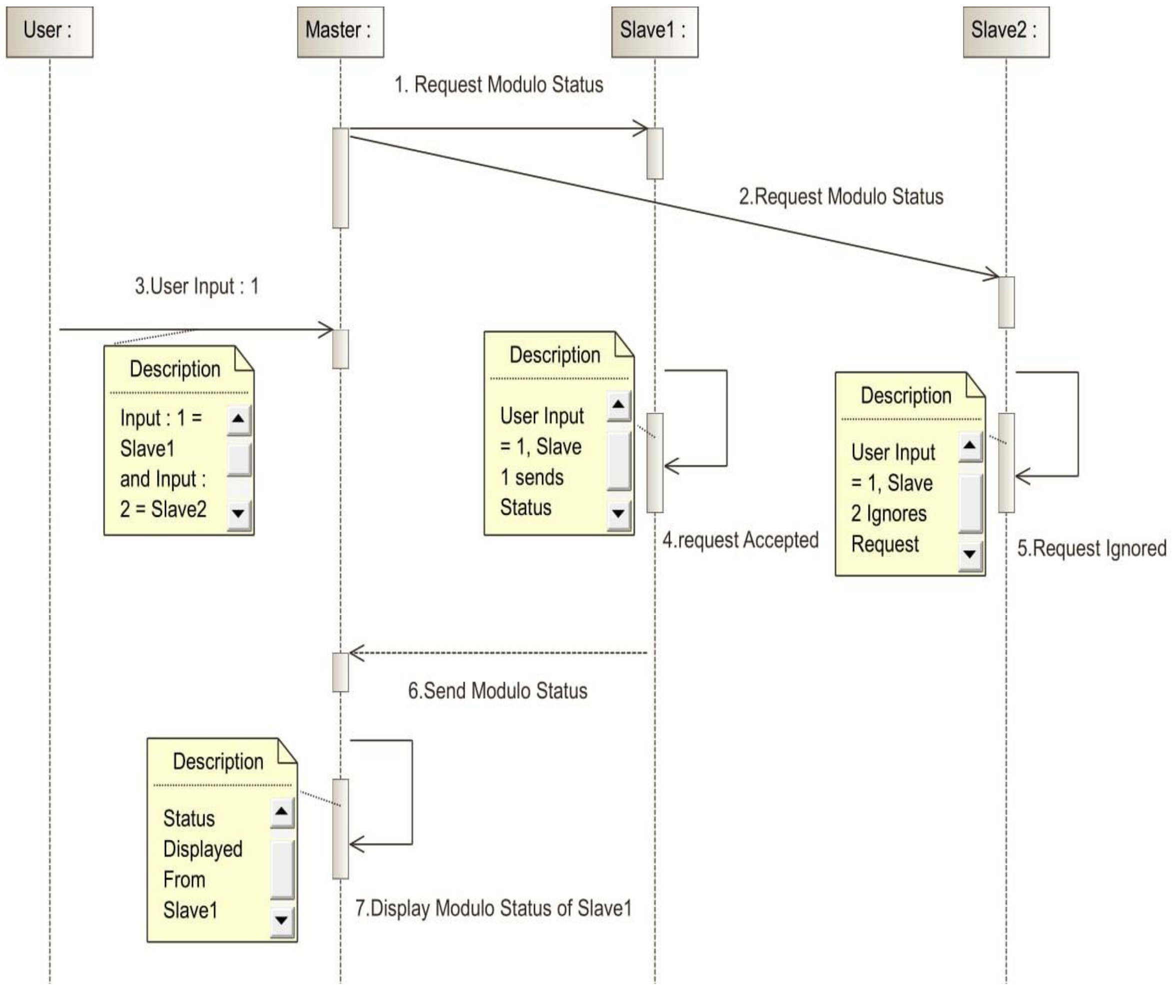


Figure 8 Sequence Diagram for Modulo Status Matrix

Chapter VI

6. Arithmetic operations

Here, the master will be sending a request to slave to perform an arithmetic operation. The master will be first executed for two things, one for sending the slave ID and the other for sending the arithmetic expression to perform required operation such as addition, subtraction, multiplication and division.

In our example, initially master sends the slave ID to all the associated slaves to confirm the presence of the required slave. Once it is confirmed, the master sends an arithmetic expression which has two integer values with a dedicated operand to the confirmed slave ID. The slave receives the expression from master in the form of input and performs a required arithmetic calculation based on the operand present in the expression. The result or the calculated value will be sent back to master for further usage.

Following code snippet will explain the functionality of arithmetic operations in the slave.

MASTER CODE

```
#include "RS485_protocol.h"
#include <SoftwareSerial.h>
const byte ENABLE_PIN = 2;
//const byte ENABLE_PIN1 = 3;
SoftwareSerial rs485 (10, 11); // receive pin, transmit pin

char input[10];
char res[10];
int charsRead = 0;
char sl;
boolean received = false;
boolean sendNow = false;

void setup()
{

  Serial.begin(9600);
  rs485.begin (28800);
  pinMode (ENABLE_PIN, OUTPUT);

}

void loop()
{
  Serial.flush();
  if (Serial.available() == 0)
  {
}
//delay(200);
while (Serial.available() > 0)
{

  charsRead = Serial.readBytesUntil('\n', input, sizeof(input) - 1);

  if (input[0] == 's')
  {
    sl = input[1];
    Serial.println(sl);
  }
  else
  {
    if (sl > 0)
    {
      input[charsRead] = sl;
      input[charsRead + 1] = '\0';
      Serial.println(input);
      sendNow = true;
      sl = NULL;
    }
  }

}

if (sendNow) {

  digitalWrite (ENABLE_PIN, HIGH);  // enable sending
  //rs485.write(sl);
  Serial.println("sending");
  // enable sending
  rs485.write(input);
  digitalWrite (ENABLE_PIN, LOW);
  sendNow = false;
  charsRead = 0;
  input[0] = 0;
}

int i = 0;
delay(1000);
digitalWrite (ENABLE_PIN, LOW);
while (rs485.available() && !sendNow) {

  char c;
```

```
    c = rs485.read();
    res[i++] = c;
    // delay(100);
    received = true;
  }
  digitalWrite (ENABLE_PIN, HIGH);
  if (i > 0)
    Serial.println(res);
}
```

Table 9 Master code for Arithmetic Operation

SLAVE1 CODE

```
#include "RS485_protocol.h"
#include <SoftwareSerial.h>
const byte ENABLE_PIN = 2;
SoftwareSerial rs485 (10, 11);  // receive pin, transmit pin

boolean replay = false;
int z = -1;
void setup()
{
  Serial.begin(9600);
  //delay(2000);
  rs485.begin (28800);
  pinMode (ENABLE_PIN, OUTPUT); // register event

}

void loop()
{

  char res[10] = {0};
  char a[5] = {0};
  char b[5] = {0};
  char operation;
  int i = 0;
  int d = 0;
  char c;
  boolean op2 = false;
  int id = 1;

  //delay(10);
```

```
digitalWrite (ENABLE_PIN, LOW);
while (rs485.available()) {

  c = rs485.read();
  Serial.println(c);
  if (c == ';')
  {
    Serial.println("End Received from slave:");
    d = rs485.read() - '0';
    Serial.println(d);
    if (d != id)
    {
      Serial.println("Wrong Slave. Exit");
      digitalWrite (ENABLE_PIN, HIGH);
      return;
    }
    replay = true;
    break;
  }

  if (c == '+' || c == '-' || c == '*' || c == '/' || c == '(' || c == ')')
  {
    i = 0;
    op2 = true;
    operation = c;
    continue;
  }

  if (op2)
    b[i++] = c;
  else
    a[i++] = c;

  // delay(20);
}

digitalWrite (ENABLE_PIN, HIGH);
int x = atoi(a);
int y = atoi(b);

//Serial.println(x);
//Serial.println(y);
switch (operation)
{
  case '+':  z = x + y;
```

```
      break;
    case '-': z = x - y;
      break;
    case '*': z = x * y;
      break;
    case '/': z = x / y;
      break;
    case '(': z = max(x, y);
      break;
    case ')': z = min(x, y);
      break;
  }


  //Serial.println(z);
  if (z > 0) {
    itoa(z, res, 10);
    //Serial.println(res);

  }

  if (replay && d == id && z > 0) {
    Serial.println("Sending result");
    Serial.println(res);
    digitalWrite (ENABLE_PIN, HIGH);
    rs485.write(res);
    digitalWrite (ENABLE_PIN, LOW);
    replay = false;
    z = -1;
  }

}
```

Table 10 Slave 1 code for Arithmetic Operation

SLAVE2 CODE

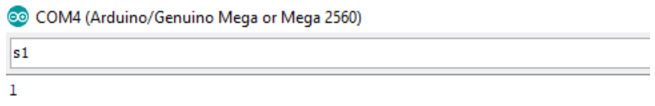
```
#include "RS485_protocol.h"
#include <SoftwareSerial.h>
const byte ENABLE_PIN = 2;
SoftwareSerial rs485 (10, 11);  // receive pin, transmit pin

boolean replay = false;
int z = -1;
void setup()
{
  Serial.begin(9600);
  //delay(2000);
  rs485.begin (28800);
  pinMode (ENABLE_PIN, OUTPUT); // register event

}

void loop()
{

  char res[10] = {0};
  char a[5] = {0};
  char b[5] = {0};
  char operation;
  int i = 0;
  int d = 0;
  char c;
  boolean op2 = false;
  int id = 2;


  digitalWrite (ENABLE_PIN, LOW);
  while (rs485.available()) {

    c = rs485.read();
    Serial.println(c);

    if (c == ';')
    {
      Serial.println("End Received from slave:");
      d = rs485.read() - '0';
      Serial.println(d);
      if (d != id)
      {
        Serial.println("Wrong Slave. Exit");
        digitalWrite (ENABLE_PIN, HIGH);
        return;
      }

      replay = true;
      break;
    }

    if (c == '+' || c == '-' || c == '*' || c == '/' || c == '(' || c == ')')
    {
      i = 0;
      op2 = true;
    operation = c;
    continue;
   }

  if (op2)
    b[i++] = c;
  else
    a[i++] = c;

  //delay(20);
 }

 digitalWrite (ENABLE_PIN, HIGH);
 int x = atoi(a);
 int y = atoi(b);

 //Serial.println(x);
 //Serial.println(y);
 switch (operation)
 {
  case '+':  z = x + y;
    break;
  case '-': z = x - y;
    break;
  case '*': z = x * y;
    break;
  case '/': z = x / y;
    break;
  case '(': z = max(x, y);
    break;
  case ')': z = min(x, y);
    break;
 }


 //Serial.println(z);
 if (z > 0) {
  itoa(z, res, 10);
  //Serial.println(res);

 }

 if (replay && d == id && z > 0) {
  Serial.println("Sending result");
  Serial.println(res);
  digitalWrite (ENABLE_PIN, HIGH);
```

```
   rs485.write(res);
   digitalWrite (ENABLE_PIN, LOW);
   replay = false;
   z = -1;
  }

}
```

Table 11 Slave 2 code for Arithmetic Operation

6.1 Execution:

1. Master will address slaves with a slave ID by sending a character value by sending the slave ID to all the slaves. For example, considering the below output window slave S1 is addressed. The master sends a character 'S1' as shown in the below output window.



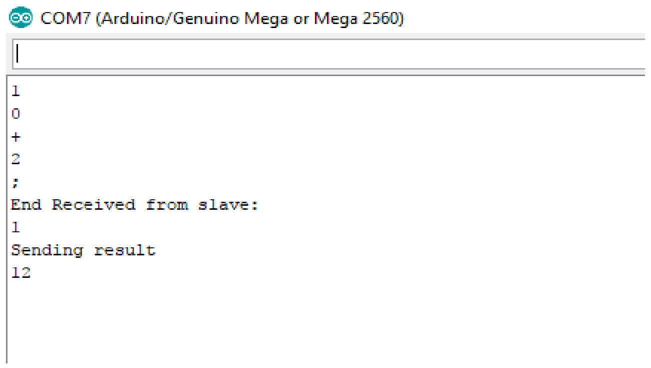
COM4 (Arduino/Genuino Mega or Mega 2560)

```
s1
```

```
1
```

Figure 9 Output for Master – Arithmetic Operation

2. Once the Master sends the slave ID, the slave's ID which is matched with the requested ID from the master will confirm its presence to the master.
3. Once the slave confirms its presence, the master will then send an arithmetic expression "10+2;" The slave receives the expression from the master in the form of an input and performs the operation based on the operand present in the expression.

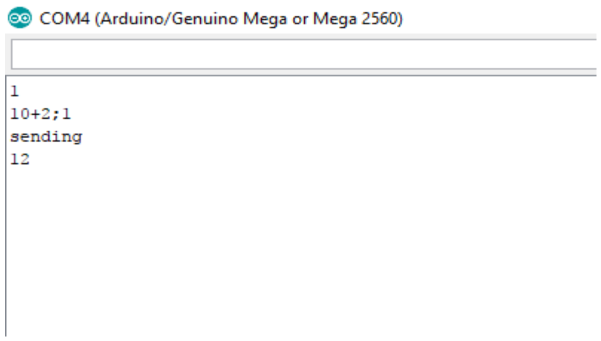


COM7 (Arduino/Genuino Mega or Mega 2560)

```
1
0
+
2
;
End Received from slave:
1
Sending result
12
```

Figure 10 Output for Slave 1 – Arithmetic Operation

4. The calculate result is then sent back to the master and this is being printed on the serial monitor of the master as shown in the below output window.

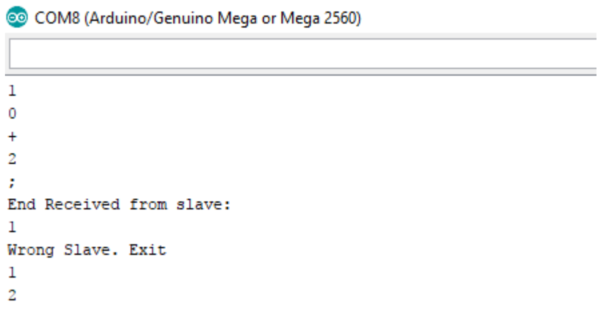


COM4 (Arduino/Genuino Mega or Mega 2560)

```
1
10+2;1
sending
12
```

Figure 11 Output for Master after slave 1 sends value– Arithmetic Operation

5. Note that, when the slave 1 is being addressed by the user the slave 2 will not perform any operation and the below output is seen on the slave 2 output window.



COM8 (Arduino/Genuino Mega or Mega 2560)

```
1
0
+
2
;
End Received from slave:
1
Wrong Slave. Exit
1
2
```

Figure 12 Output for slave 2 – Arithmetic Operation

6.2 Sequence Diagram:

The below Figure represents the sequence diagram for Arithmetic operations between Master and Slaves. Here, Master is sending request for Arithmetic task to each Slave. Every Slave is capable to perform different operations such as addition, subtraction, multiplication, division, but selection of specific Slave is done by User by using Serial Monitor of Master. If User input = s1 that means Slave1 is being selected for operation and rest of all Slaves are being ignored for that time. According to the example User sends two values 10 and 2 to the Slave1 for addition, So Slave1 is adding those two values and sends result back to the Master.

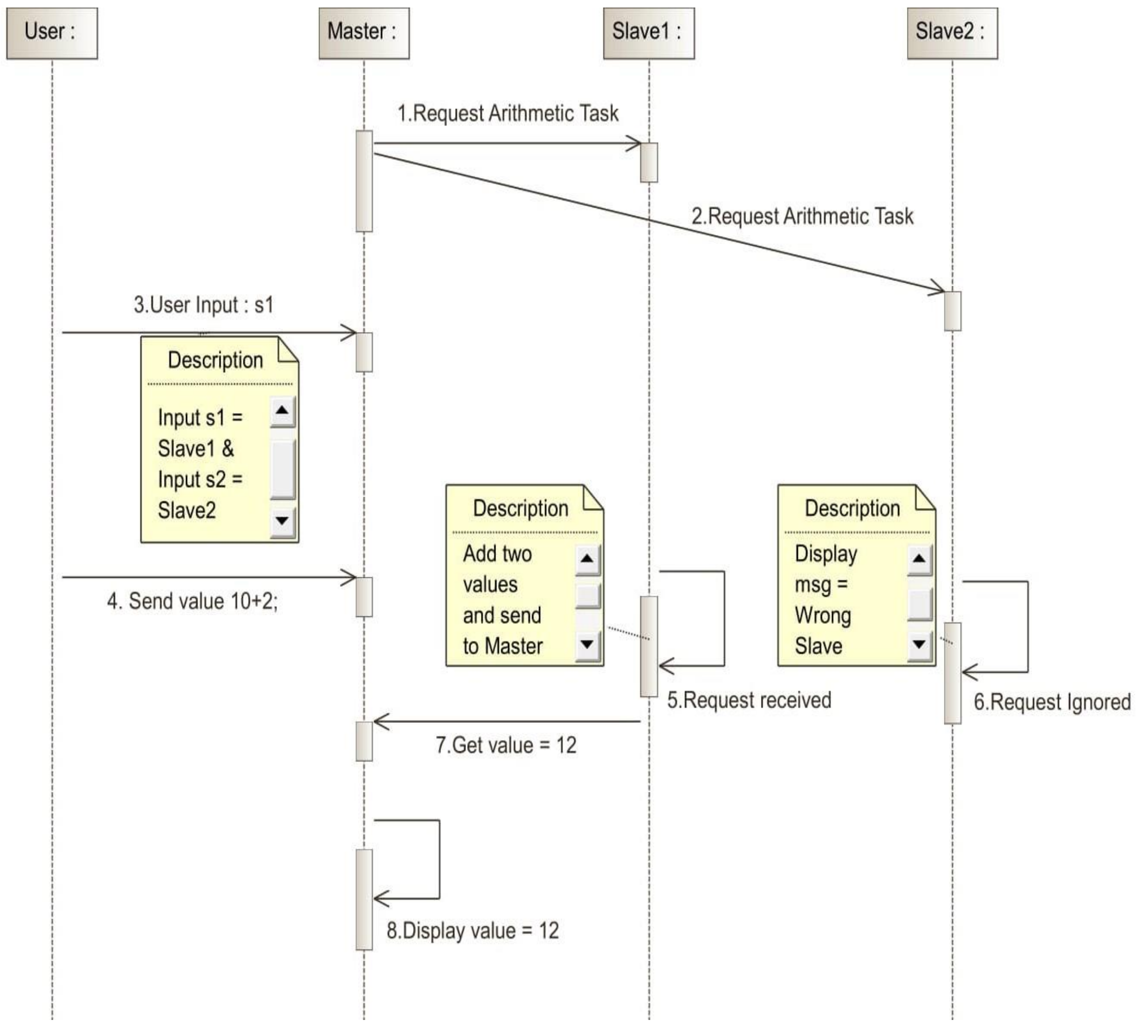


Figure 13 Sequence Diagram – Arithmetic Operation

Chapter VII

7. Polling

In this scenario, the master repeatedly polls slaves in a cyclic manner. The master sends a request to each slave and when slaves receive the request, they respond with a value to the master. Since the request will be send to all slaves at the same time, this may lead to a collision.

In order to avoid collision, the master sends request with the slave address to all slaves, and slaves will only respond back to the master if it is being addressed to. Finally, when the master receives the response sent from all slaves, it sends back the respond message (Slave address with value) to all slaves.

Initial setup: Master with two slaves connected with rs485 Modbus.

MASTER CODE

```
#include "RS485_protocol.h"
#include <SoftwareSerial.h>
#define TOTAL_SLAVES 2 // Total slaves presently in the network
const byte ENABLE_PIN = 2; //output pin
long data_matrix[TOTAL_SLAVES][2]; //matrix to send back to the slaves from master
SoftwareSerial rs485 (10, 11); // Receive and Transmit pins

boolean stopp = false;
int n = 1;
uint8_t ch;
int col = 0;
long  bignum;
uint8_t b[4];

void setup() {

  rs485.begin (28800);
  pinMode (ENABLE_PIN, OUTPUT);
  Serial.begin(9600);

}


void loop() {

    Serial.flush();
    char req[2];
    req[0] = 'R'; //Request messgae
    req[1] = n + '0'; //Request message with Slave address
    digitalWrite (ENABLE_PIN, HIGH); //begin transmission
    rs485.write(req, sizeof(req)); //sending request message
    Serial.println("requesting from slave ");
    Serial.println(n);
    digitalWrite (ENABLE_PIN, LOW); //end transmission
Serial.println(previousMillis);
  delay(1000);

  while ( rs485.available())
  {
    digitalWrite (ENABLE_PIN, LOW); //begin receive
    while ((ch = rs485.read()) != 'E') { //if receive is value and not End

      b[0] = ch;
      b[1] = rs485.read();
      b[2] = rs485.read();
      b[3] = rs485.read();

      bignum = *((long *)b); //reading long values from slave
      Serial.println("Value received:");
      data_matrix[n - 1][col++] = bignum;
      Serial.println(bignum);
      delay(1000);

    }
    if (ch == 'E') { //if receive is end

      Serial.println("End Value received from slave:");
      Serial.println(n);

      n++; //continue with next slave
      col = 0;

      digitalWrite (ENABLE_PIN, HIGH); //begin transmission
      delay(1000);
      ch = 0;
      if (n > TOTAL_SLAVES) {
        Serial.println("Sending back all values");
        for (int i = 0; i < TOTAL_SLAVES; i++)
        {
          uint8_t buf[6] = {0};
          for (int j = 0; j < 2; j++) //change j<2 if values receiving are higher than 2
          {
            Serial.println(data_matrix[i][j]);
            buf[5] = (uint8_t)((data_matrix[i][j] >> 24) & 0xFF);
            buf[4] = (uint8_t)((data_matrix[i][j] >> 16) & 0xFF);
            buf[3] = (uint8_t)((data_matrix[i][j] >> 8) & 0xFF);
          buf[2] = (uint8_t)(data_matrix[i][j] & 0xFF);
          buf[1] = i + 1; //slave address stored in this buffer
          buf[0] = 'S'; //S : slave
          rs485.write(buf, sizeof(buf)); //send all the values with slave address to all slaves
          delay(100);
         }
       }
      n = 1; //slave address back to 1, repeat from the start
     }

     digitalWrite (ENABLE_PIN, LOW); //end transmission

   }
  }

}
```

Table 12 Master code for Polling

Slave 1 Code:

```
#include "RS485_protocol.h"
#include <SoftwareSerial.h>
const byte ENABLE_PIN = 2; //output pin
SoftwareSerial rs485 (10, 11); //receive and transmit pins

boolean replay = true;
boolean requested = false;
int id = 1; //slave address
uint8_t ch;

void setup()
{

  randomSeed(analogRead(0));
  Serial.begin(9600);
  rs485.begin (28800);
  pinMode (ENABLE_PIN, OUTPUT);
}

void loop()
{
```

```
digitalWrite (ENABLE_PIN, LOW); //begin receive
while (rs485.available())
{
 ch = rs485.read();

 if (ch == 'R')
 {
  //we are recieiving request from master

  if (rs485.read() != '1') //check if request is R1
   return;
  Serial.println("Request received");
  requested = true;
 }
 else
 {

  uint8_t b[4];
  if (ch == 'S') {
   int slave_num = rs485.read();

   b[0] = rs485.read();
   b[1] = rs485.read();
   b[2] = rs485.read();
   b[3] = rs485.read();
   long  bignum = *((long *)b);
   Serial.println("Got Value from Slave:");
   Serial.println(slave_num);
   Serial.println("Value:");
   Serial.println(bignum); //printing values from master
  }
 }
 delay(100);
}

if (requested)
{
 digitalWrite (ENABLE_PIN, HIGH);
 //rs485.write(x);
 long n = 837426840;
 uint8_t buf[4];
 for (int i = 0; i < 2; i++) { //change i<2 to higher value is value to be sent is more than 2
  buf[3] = (uint8_t)((n >> 24) & 0xFF);
  buf[2] = (uint8_t)((n >> 16) & 0xFF);
  buf[1] = (uint8_t)((n >> 8) & 0xFF);
  buf[0] = (uint8_t)(n & 0xFF);
```

```
      Serial.println("Sending to Master");
      Serial.println(n);
      rs485.write(buf, sizeof(buf)); //send long values to master when request is received
      n = n + 100000000;
      delay(10);
    }

    rs485.write('E'); //send End to master so as slave 1's work is done, the master can
continue to slave 2
    digitalWrite (ENABLE_PIN, LOW);
    requested = false;

  }
  delay(100);
}
```

Table 13 Slave 1 code for Polling

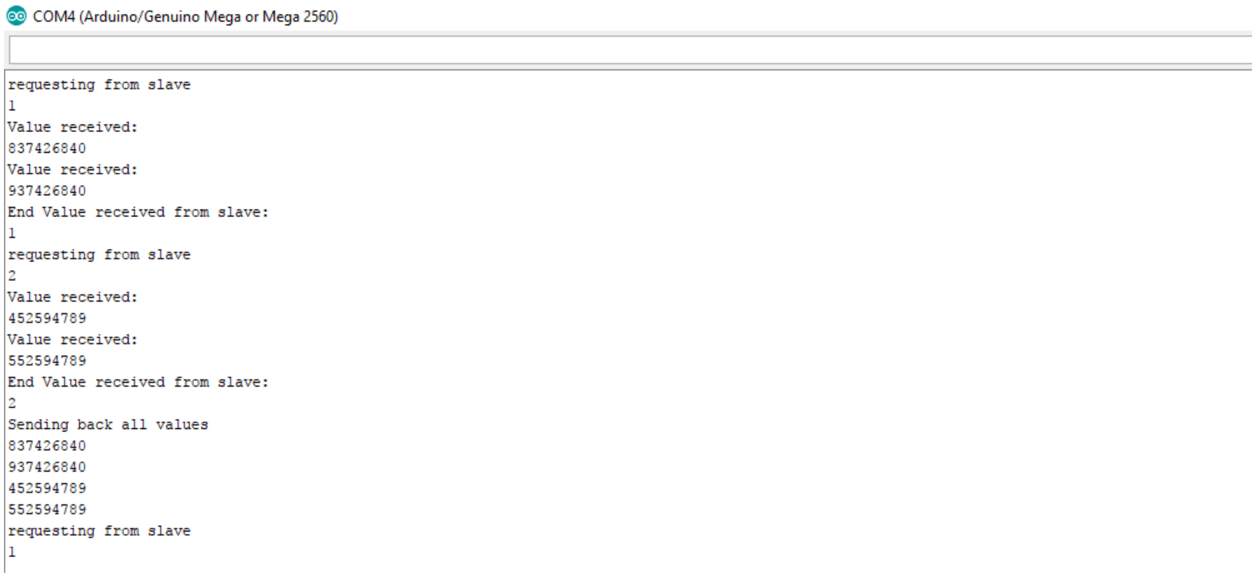
Slave 2:

```
#include "RS485_protocol.h"
#include <SoftwareSerial.h>
const byte ENABLE_PIN = 2; //output pin
SoftwareSerial rs485 (10, 11); //receive and transmit pins

boolean replay = true;
boolean requested = false;
int id = 2; //slave address
uint8_t ch;

void setup()
{

  randomSeed(analogRead(0));
  Serial.begin(9600);
  rs485.begin (28800);
  pinMode (ENABLE_PIN, OUTPUT);
}

void loop()
{
  digitalWrite (ENABLE_PIN, LOW); //begin receive
  while (rs485.available())
  {
    ch = rs485.read();
    if (ch == 'R')
    {
      //we are recieiving request from master

      if (rs485.read() != '2') //check if request is R1
        return;
      Serial.println("Request received");
      requested = true;
    }
    else
    {

      uint8_t b[4];
      if (ch == 'S') {
        int slave_num = rs485.read();

        b[0] = rs485.read();
        b[1] = rs485.read();
        b[2] = rs485.read();
        b[3] = rs485.read();
        long  bignum = *((long *)b);
        Serial.println("Got Value from Slave:");
        Serial.println(slave_num);
        Serial.println("Value:");
        Serial.println(bignum); //printing values from master
      }
    }
    delay(100);
  }

  if (requested)
  {
    digitalWrite (ENABLE_PIN, HIGH);
    //rs485.write(x);
   long n = 452594789;
    uint8_t buf[4];
    for (int i = 0; i < 2; i++) { //change i<2 to higher value is value to be sent is more than 2
      buf[3] = (uint8_t)((n >> 24) & 0xFF);
      buf[2] = (uint8_t)((n >> 16) & 0xFF);
      buf[1] = (uint8_t)((n >> 8) & 0xFF);
      buf[0] = (uint8_t)(n & 0xFF);
      Serial.println("Sending to Master");
      Serial.println(n);
      rs485.write(buf, sizeof(buf)); //send long values to master when request is received
      n = n + 100000000;
    delay(10);
   }

   rs485.write('E'); //send End to master so as slave 2’s work is done; the master can continue
to slave 1
   digitalWrite (ENABLE_PIN, LOW);
   requested = false;

  }
  delay(100);
}
```

Table 14 Slave 2 code for Polling

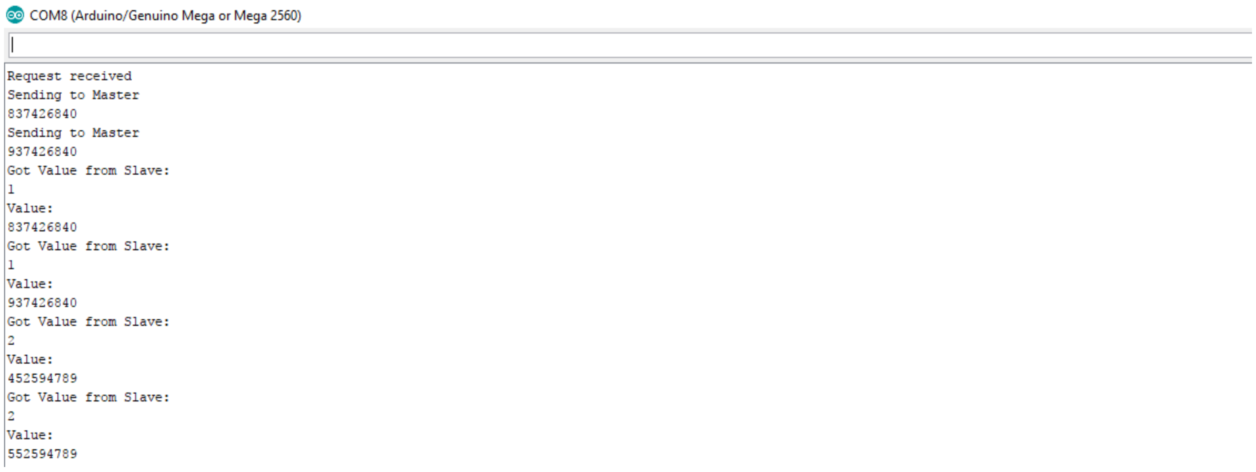
The working steps are mentioned below:

(a) The master cyclically requests slaves to send their data. The master sends request R1 (Request and slave 1 address) to all slaves in the network.
(b) The slaves on receiving the request checks if the assigned address matches to its ID. If not, then it waits for the master to send next request. If it matches with the ID, the slave sends two long values along with character E (indicating end, the slave has no more data to send) to the master.
(c) The master then receives the values from slave 1 and moves on to R2 (Request and slave 2 address).
(d) Similarly, as the (b) step, the slave 2 sends its long values to the master.
(e) On receiving all the values from both slaves on the master, the master sends back the received values to all slaves with slave address.
(f) The master then repeats step (a) for nth time iteratively.



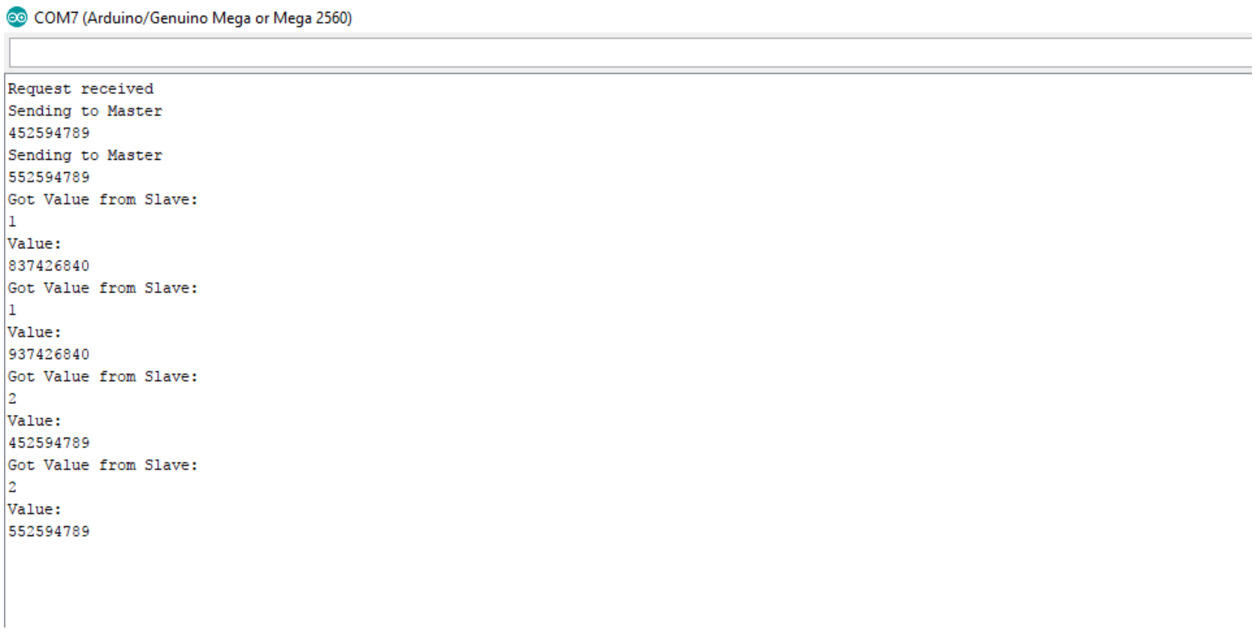
```
requesting from slave
1
Value received:
837426840
Value received:
937426840
End Value received from slave:
1
requesting from slave
2
Value received:
452594789
Value received:
552594789
End Value received from slave:
2
Sending back all values
837426840
937426840
452594789
552594789
requesting from slave
1
```

Figure 14 Output for Master – Polling


COM8 (Arduino/Genuino Mega or Mega 2560)

```
Request received
Sending to Master
837426840
Sending to Master
937426840
Got Value from Slave:
1
Value:
837426840
Got Value from Slave:
1
Value:
937426840
Got Value from Slave:
2
Value:
452594789
Got Value from Slave:
2
Value:
552594789
```

Figure 15 Output for Slave 1 – Polling


COM7 (Arduino/Genuino Mega or Mega 2560)

```
Request received
Sending to Master
452594789
Sending to Master
552594789
Got Value from Slave:
1
Value:
837426840
Got Value from Slave:
1
Value:
937426840
Got Value from Slave:
2
Value:
452594789
Got Value from Slave:
2
Value:
552594789
```

Figure 16 Output for Slave 2 – Polling

Since master repeatedly requests slaves to send their values, there could be an instance when a specific slave is disconnected from the network and won't be able to reply to the master. Here the master will be adhered on slave's reply and would not continue to the next slave. In such a situation, we include concept of millis(). Millis() is a time function which returns the number of milliseconds since the Arduino board began execution of the current program.

We assign timeout with a value in milliseconds, and compare it with millis(). The master shall wait for the non-responsive slave to respond. In that amount of time, if the timeout completes or exceeds the millis(), the master shall exit and move to next slave and continue henceforth.

```
void loop() {

  unsigned long previousMillis = 0;
  long interval = 600; //interval time
  unsigned long currentMillis = millis(); //current time of execution
  if (currentMillis - previousMillis > interval) { //comparison
    previousMillis = currentMillis;
    Serial.flush();
    char req[2];
    req[0] = 'R';
    req[1] = n + '0';
    digitalWrite (ENABLE_PIN, HIGH);
    rs485.write(req, sizeof(req));
    Serial.println("requesting from slave ");
    Serial.println(n);
    digitalWrite (ENABLE_PIN, LOW);
    Serial.println(previousMillis);
    while (previousMillis > 10000) //if current slave is non-responsive
    {
      n = 1; //master moves to next slave
      return;
    }
  }
```

Table 15 Addition of Millis() in Master code for Polling

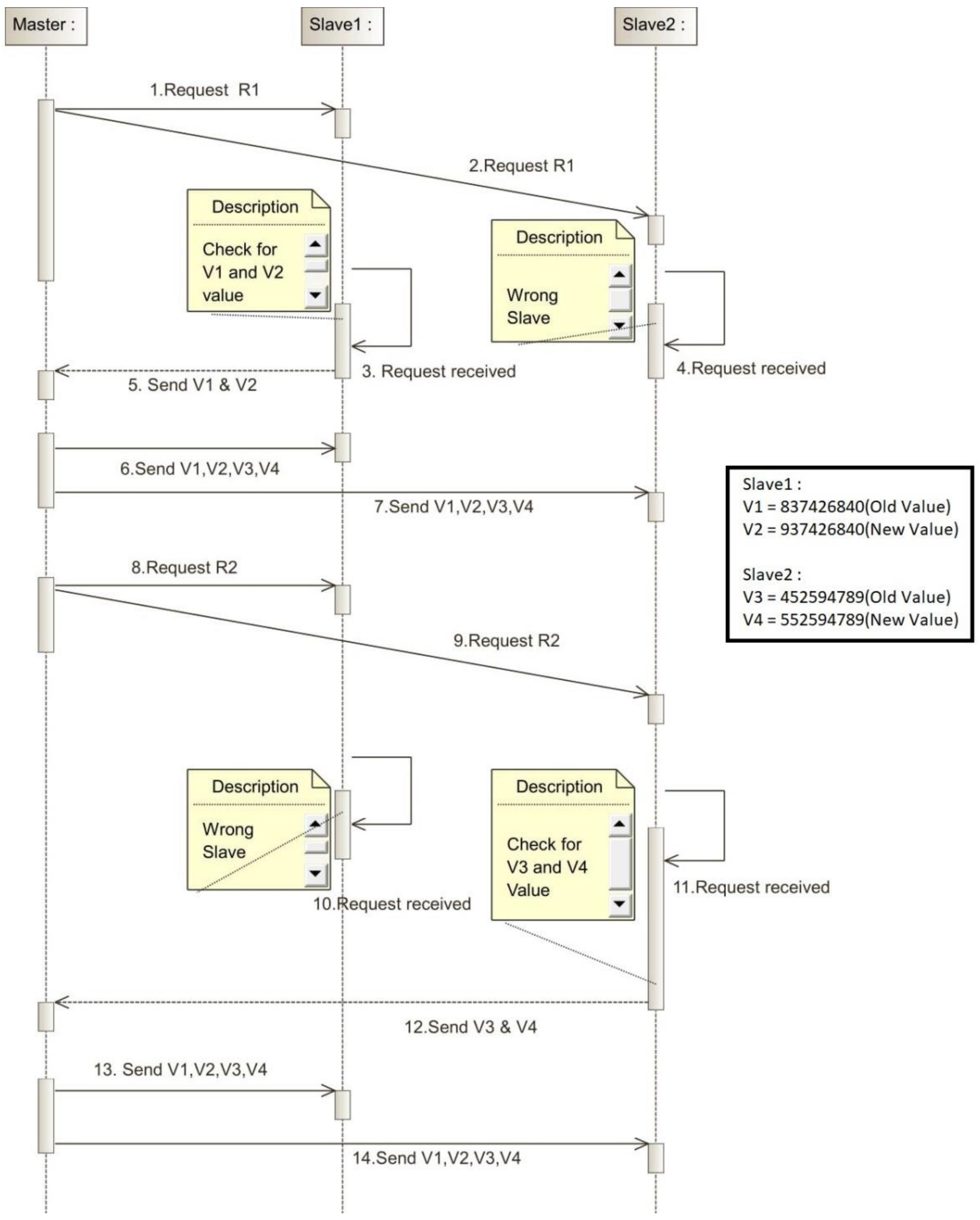


Figure 17 Sequence Diagram – Polling

Chapter VIII

8. Web Interface

This section describes to store the values onto MySQL table while Arduino program is executing. The values will be inserted/updated based on the primary key values and deleted based on the user request. To perform web interface, we make use of Raspberry Pi. From the above objectives, we will be using Distance Matrix and Arithmetic operations to demonstrate storing values from Arduinos and to retrieve values from MySQL table to Arduino.

8.1 Raspberry Pi

It is a pocket-sized operating system which has an SD card slot that acts as a memory. It uses Raspbian as its operating system which is a version of Linux built specifically for Raspberry Pi. It has USB, micro USB, HDMI and Ethernet slots which can be used to connect to a laptop, desktop or even a monitor.

To connect to a monitor, only HDMI cable will be sufficient. But when the user wants to connect to a laptop or a desktop, the following steps need to be followed:

(a) The initial required materials along with Raspberry Pi that would be needed are: Ethernet cable, Laptop, Micro USB cable and SD card with Raspbian installed.
(b) To connect Raspberry Pi to laptop, ethernet cable is a must as it provides GUI (Graphical User Interface) which can be viewed through laptop display. The ethernet connection should be as good as 100Mbps speed. To bridge the connection between laptop and Raspberry Pi, a software called VNC viewer needs to be installed. Through this we can access Raspberry Pi remotely.
(c) After setting SD card with Raspbian pre-installed onto Raspberry Pi (from Raspberry Pi website; see references for link), we connect our micro USB cable from laptop to Raspberry Pi to power it up.
(d) The next step is to share internet to Raspberry Pi over ethernet. On the laptop, go to Network and Sharing Center. Click on the Wi-Fi network the laptop is connected to. Click on properties, go to Sharing and click on “Allow other users to connect” and select “Ethernet”. Now back to Network and Sharing Center, click on the “Ethernet” link created and click on Properties and double click on “Internet Protocol Version (IPv4). Here it will be given that the current laptop’s IP is assigned to 192.168.137.1. That means 192.168.137.x, the ‘x’ will be in a range of 1-255, and Raspberry Pi will be connected to one of the ranges.
(e) To see which IP address the Raspberry Pi is connected to, the user needs to use a software to scans IP’s the Wi-Fi is connected to. We have made use of “Advanced IP Scanner” (see references for link). When the user scans the range of IP’s from 192.168.137.1-192.168.137.255, it will show a list of connected devices along with Raspberry Pi.

(f) The final step is for GUI. VNC server/viewer provides/bridges the connection between laptop and Raspberry Pi. It acts as a virtual machine for Raspbian. Initially we need an SSH server to request connection. To do this we install Putty (see references for link) and via SSH we can connect to Raspberry Pi. Once the Putty is open, it will ask for IP address, the IP address of Raspberry Pi can be found out from "Advanced IP Scanner" as mentioned on (e) step. Once entered, it will open an SSH (command prompt) and will ask for user and password. The default user is "pi" and password is "raspberry". Then enter the following command:

```
$ vncserver :1
```

Table 16 Putty configuration

This will create an instance on VNC viewer and will accept the request to access. Download and install VNC client (see references for link). Open VNC viewer, enter IP address of Raspberry Pi (IP address and :1 should be added along with it). An instance will be created on VNC viewer. Open the instance and you will be directed to the homepage of Raspbian.

8.2 Installing PhpMyAdmin, Apache and MySQL server

Instead of connecting Arduinos to laptop, we connect Arduinos to Raspberry Pi which will now be our operating system. The first step is to access PhpMyAdmin page. To do this, we install MySQL server, apache and PhpMyAdmin. We enter the commands on the "Terminal" of Raspbian.
To make sure the Raspbian is up to date, we enter the following command:

```
$ sudo apt-get update
```

Table 17 Raspbian update configuration

The next step is to install apache and its libraries, hereby the following:

```
$ sudo bash
apt-get install apache2 apache2-doc apache2-utils
apt-get install libapache2-mod-php5 php-pear php5-xcache
```

Table 18 Apache installation

The next step is to install database connectivity and MySQL server:

```
apt-get install mysql-server mysql-client
```

Table 19 MySQL Server installation

Once, the above command is executed, it will ask the user to set a password for the user – ‘root’. Once entered, it will be prompted and confirmed on a blue screen.
Final step is to install PhpMyAdmin on Raspbian. The commands are:

```
apt-get install phpmyadmin
```

Table 20 PhpMyAdmin installation

The package will begin installing and will ask the user which web server needs to be installed, choose “apache2”. It will then ask to configure the database, click “Yes” and finally it will be asked to enter administrative password, this password is same as entered during mysql-server installation.
Finally, we configure Apache to work with PhpMyAdmin. Enter the following command:

```
nano /etc/apache2/apache2.conf
```

Table 21 Apache with PhpMyAdmin configuration

Navigate to the bottom of the file and this add this:

```
Include /etc/phpmyadmin/apache.conf
```

Table 22 Apache file configuration

Save the file (CTRL +X and enter Y when asked to save). Restart Apache2:

```
/etc/init.d/apache2 restart
```

Table 23 Restart Apache configuration

Once this is done, the PhpMyAdmin will be ready to access. To access the page, open browser, and type “localhost/phpmyadmin” and press enter.

8.3 Distance

As mentioned under “Distance Matrix” objectives, master will request the slaves to send their distance values between each slave. The request (slave ID) will be provided by the user on Master terminal and the distances will be sent by the requested slave. The values gained will be stored onto the database table.

The first step is to create a database onto PhpMyAdmin page. To create a database and to make sure it will be remotely accessed (accessed by another local machine which is connected to same network as the host), we follow the following steps:

(a) We need to connect to the SQL server through the Terminal of Raspbian. On the terminal, we give the following command:

```
$ mysql -u root -p
```

Table 24 Connection to MySQL Server

Here, u: username (that is root) and p: password

(b) Creating a database Distance for Distance Matrix. The following command:

```
CREATE DATABASE Distance;
```

Table 25 Database Creation

(c) To make sure the database will be accessed by another machine, we use GRANT order. With this, the external user can make changes only on the database Distance and not to any other. The command is:

```
GRANT ALL PRIVILEGES ON Distance.* TO user@’%’ IDENTIFIED BY
‘password’;
```

Table 26 Granting privileges to DB

Here, Distance.* indicates that all tables will be accessed under the database Distance. @% indicates all addresses connected to the same network will have the right to access the database.

(d) Once the above commands are executed, MySQL needs to apply the changes made, with the FLUSH command:

```
flush privileges;
```

Table 27 Flush privileges

There is another step to be followed only once when configuring a database. The step is to configure SQL to accept external connections. On the Terminal, the following command needs to be executed:

```
$ sudo nano /etc/mysql/my.cnf
```

Table 28 Configure external requests

There needs to be added a comment to a line which points to binding address, the line to which the comment (#) needs to be added is:

```
#bind-address = 127.0.0.1
```

```
/etc/init.d/mysql restart
```

Table 29 Binding address

The above command saves and restarts MySQL server.

Now we have remote and host access to database Distance. We can alter changes from any device. Before execution of Arduinos there should a table and attributes to add, we divide the steps into three:

(a) Creating tables and its attributes.
(b) Creating a python script to insert values and to retrieve values to/from the table
(c) Execution of Arduinos.

8.3.1 Creating tables and its attributes

A table should be created for the distance values to be inserted. Each slave will have different distance values corresponding to its neighborhood slaves. The table can be created on the console of PhpMyAdmin page or by using ‘Create table’ option on the same page. The tablecreated for the distances is as follows:

```
Create table dist (sno int(4) auto_increment unique not null, slno int(4) primary key not
null, a int(4) not null, b int(4) not null, c int(4) not null, d int(4) not null, e int(4) not
null);
```

Table 30 Creating table dist

dist: table name;

sno: serial number which increments as slave number(slno) increases;

slno: slave number with primary key;

a,b,c,d,e: corresponding distance values of slave;

8.3.1.1 Trigger

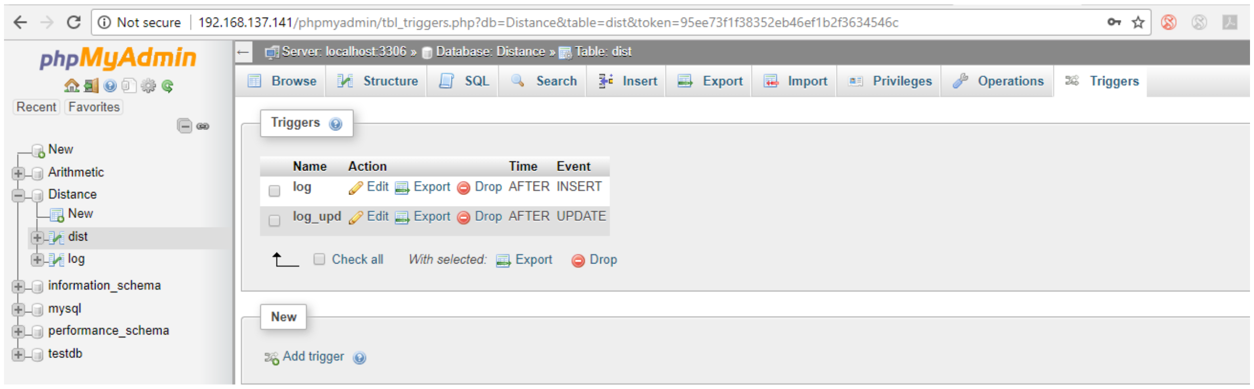
A trigger is an event that is enabled when insert/update/delete operation is performed. This trigger will be created under the table dist.

We will create another table called ‘log’ with timestamp(datetime) which will save the data inserted/updated on the main table (‘dist’). By this we will know which slave had inserted/updated its values based on the time. The first step is to create a child table called ‘log’. The attributes of this table will be the same as the parent table (‘dist’) along with a new attribute for datetime. However, this table will have no primary or unique keys, as the parent table will have the command over it. The table is as follows:

```
Create table log (sno int(4) not null, slno int(4) not null, a int(4) not null, b int(4) not null, c int(4) not null, d int(4) not null, e int(4) not null, exec_time datetime not null);
```

Table 31 Creating table log

Now, the log table needs to be linked to its parent table. To do this we add a trigger event under 'dist' table. The trigger event can be found in PhpMyAdmin page under tables.



192.168.137.141/phpmyadmin/tbl_triggers.php?db=Distance&table=dist&token=95ee73f1f38352eb46ef1b2f3634546c

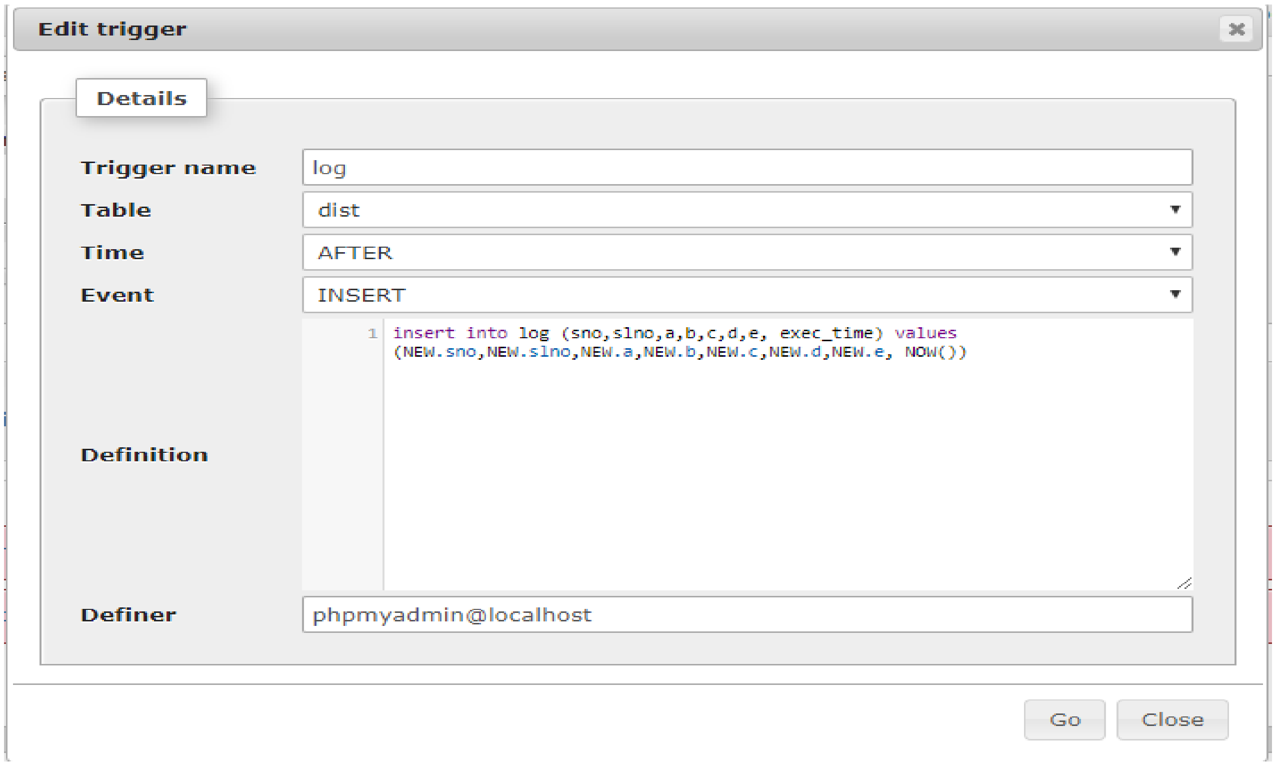
Server: localhost:3306 » Database: Distance » Table: dist

	Name	Action	Time	Event
☐	log	Edit Export Drop	AFTER	INSERT
☐	log_upd	Edit Export Drop	AFTER	UPDATE

Figure 18 Triggers -PhpMyAdmin

Two trigger events will be created for the table 'dist'. One trigger event is created for insertion of values and second trigger event is created for update of values. Whenever an insertion/update is occurred on the parent table 'dist', the trigger will be activated and the same values will be stored as a log file with time on the child table 'log'. The values in 'log' table cannot be edited but can be emptied.

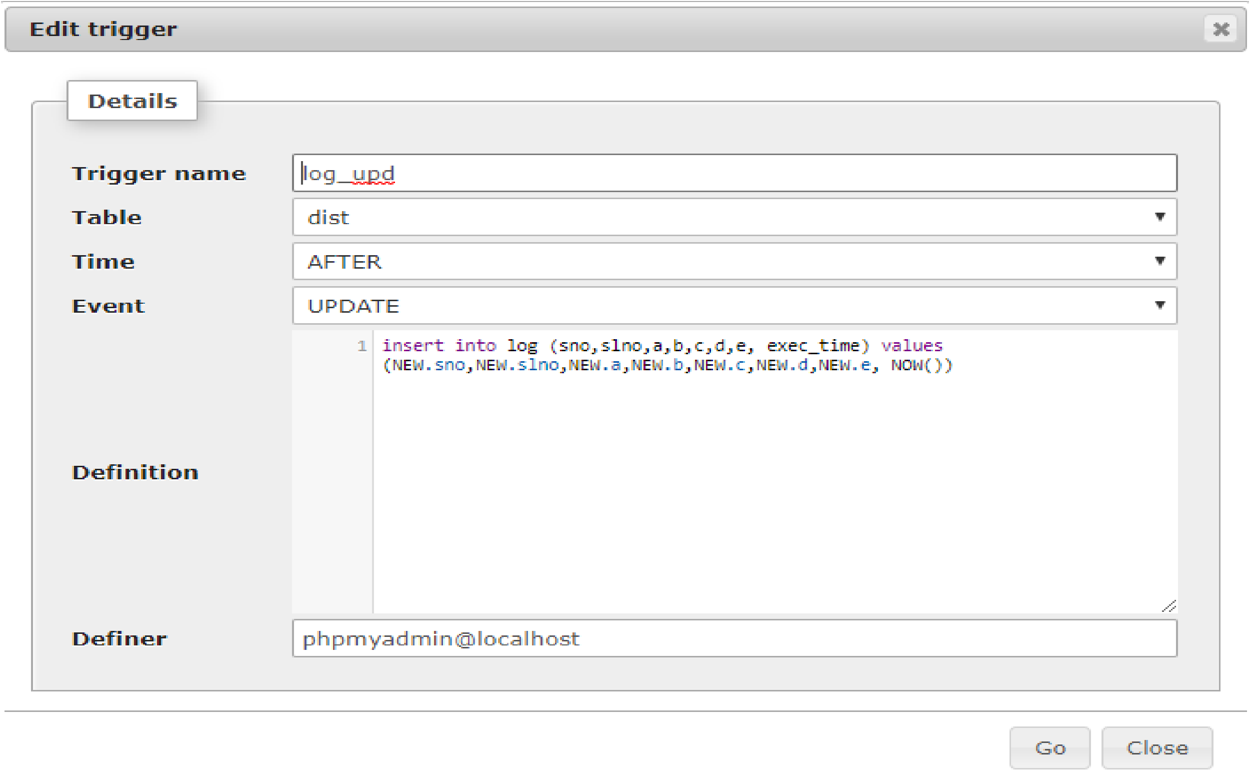
Finally, the contents of the 'dist'/'log' table will be sent to the Master Arduino so that the Master will know which slave was updated and at what time.



Trigger name	log
Table	dist
Time	AFTER
Event	INSERT
Definition	
Definer	phpmyadmin@localhost

```
insert into log (sno,slno,a,b,c,d,e, exec_time) values
(NEW.sno,NEW.slno,NEW.a,NEW.b,NEW.c,NEW.d,NEW.e, NOW())
```

Figure 19 Insert After Trigger -PhpMyAdmin



Trigger name	log_upd
Table	dist
Time	AFTER
Event	UPDATE
Definition	
Definer	phpmyadmin@localhost

```
insert into log (sno,slno,a,b,c,d,e, exec_time) values
(NEW.sno,NEW.slno,NEW.a,NEW.b,NEW.c,NEW.d,NEW.e, NOW())
```

Figure 20Update After Trigger - PhpMyAdmin

8.3.2 Creating a python script to insert values and to retrieve values to/from the table

Since we are running web interface on Raspbian operating system. The scripts that are needed to be written are in Python. These scripts are to be executed on the Terminal of Raspbian.

Two scripts are written for the distance matrix program. One script is for insertion of values onto the MySQL page and the second script is for retrieving the values from MySQL page to Master Arduino.

The first script is ‘db.py’ and it goes as:

```
import serial
import time
import MySQLdb as mdb

arduino = serial.Serial("/dev/ttyACM1")
arduino.baudrate=9600

data1 = arduino.readline()
data2 = arduino.readline()
data3 = arduino.readline()
data4 = arduino.readline()
data5 = arduino.readline()
data6 = arduino.readline()



slno = data1
a = data2
b = data3
c = data4
d = data5
e = data6


con = mdb.connect('localhost', 'phpmyadmin', 'some_pass', 'Distance');

with con:
   cursor = con.cursor()
   cursor.execute("""INSERT INTO dist VALUES("%s,%s,%s,%s,%s,%s) ON DUPLICATE
KEY    UPDATE a = VALUES(a), b = VALUES(b), c = VALUES(c), d = VALUES(d), e =
VALUES(e)""",(slno,a,b,c,d,e));
   con.commit()
   cursor.close()
```

Table 32 Python script for inserting/update values

/dev/ttyACM1: It is the port number to which the Arduino is connected to, the port number is 1.

'localhost', 'phpmyadmin', 'some_pass', 'Distance': are basically ‘IPAddress’, ’username’, ’pass’, ’database’.

Since we are performing insertion and update on the same script, the duplicate values on the primary key (slno) will be updated.

The second script is for retrieving the values from MySQL table and it is ‘save.py’. It goes as:

```
import MySQLdb as mdb
import serial

con = mdb.connect('localhost', 'phpmyadmin', 'some_pass', 'Distance');

cur = con.cursor()

cur.execute("""select * from dist""")

row = cur.fetchall()

while row is not None:

    for i in range(len(row)):
        print row[i]
        ser = serial.Serial("/dev/ttyACM0")
        ser.baudrate=9600
        ser.write(row[i])

    break

cur.close()
```

Table 33 Python script for retrieving values from server

row[i]: i is the i’th row that can be retrieved onto the master. ‘i’ can be on the range, if it is 1, it will retrieve only the first row and if it is 2, it will retrieve the first two rows and so on. Here row[i] suggests that it will retrieve all the rows in the table.

8.3.3 Execution of Arduinos

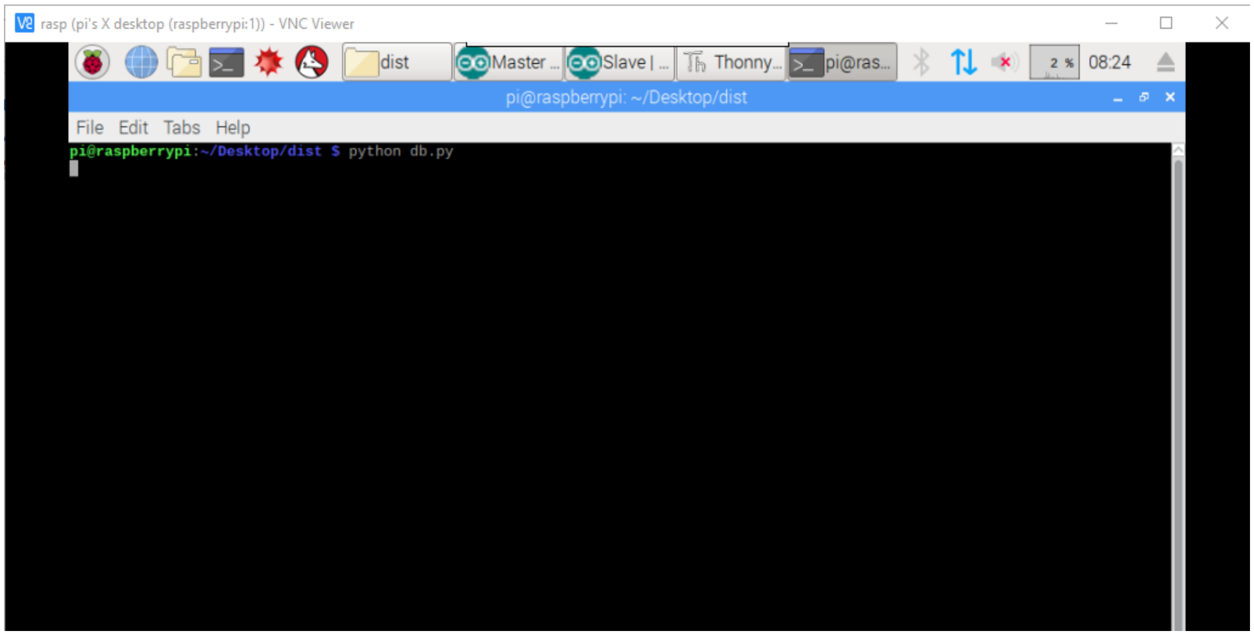
In the last step, we execute the Arduinos (Master and slaves) first and then the scripts. Once the script executes, it waits for the user to enter values on the Serial monitor of the slave. When the user enters, the distance values are inserted through Python script from the specific slave onto the MySQL page.

First, we execute the Master and Slave Arduinos. The Master serial port should match with ‘save.py’ script and the slave serial port should match with ‘db.py’ script. The Arduino codes are the same as given above under objectives section.

Secondly, we open Terminal of the Raspbian and execute db.py script.

```
python db.py
```

Table 34 Python script executing – db.py

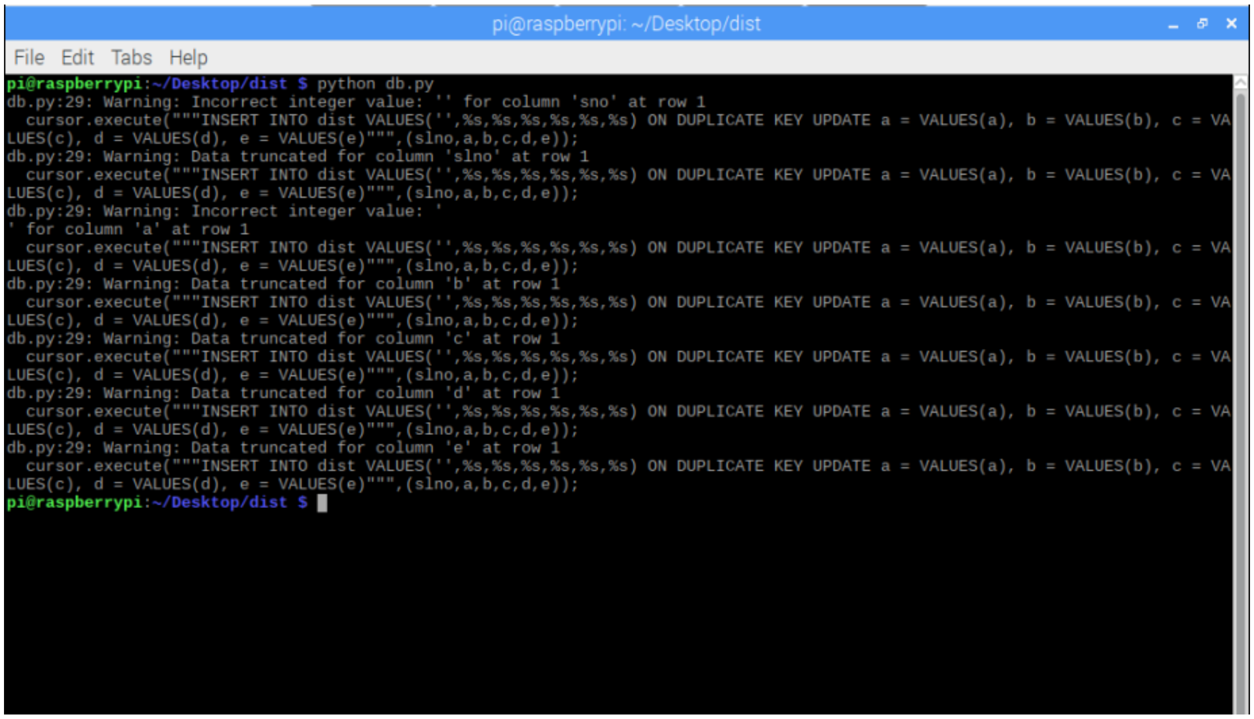


rasp (pi's X desktop (raspberrypi:1)) - VNC Viewer
dist
Master ...
Slave | ...
Thonny...
pi@ras...
08:24
pi@raspberrypi: ~/Desktop/dist
File Edit Tabs Help

```
pi@raspberrypi:~/Desktop/dist $ python db.py
```

Figure 21 Python Script – db.py – Distance Matrix

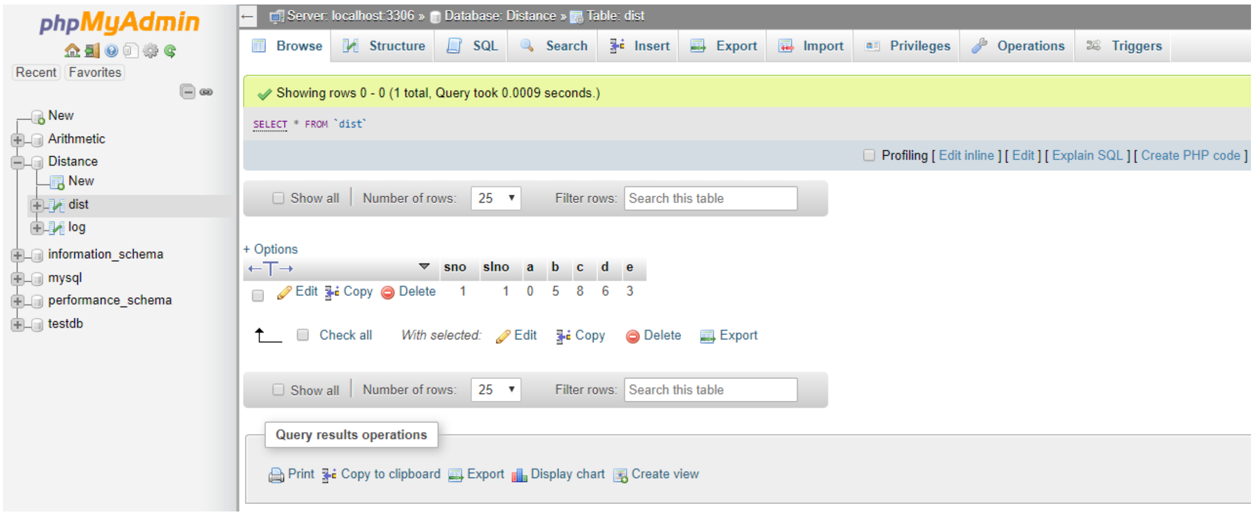
It waits for the input to be taken from the Arduino program. We then open the serial ports of master and slave and enter the corresponding slave number on the master.



```
pi@raspberrypi:~/Desktop/dist $ python db.py
db.py:29: Warning: Incorrect integer value: '' for column 'sno' at row 1
  cursor.execute("""INSERT INTO dist VALUES('',%s,%s,%s,%s,%s,%s) ON DUPLICATE KEY UPDATE a = VALUES(a), b = VALUES(b), c = VA
LUES(c), d = VALUES(d), e = VALUES(e)""",(slno,a,b,c,d,e));
db.py:29: Warning: Data truncated for column 'slno' at row 1
  cursor.execute("""INSERT INTO dist VALUES('',%s,%s,%s,%s,%s,%s) ON DUPLICATE KEY UPDATE a = VALUES(a), b = VALUES(b), c = VA
LUES(c), d = VALUES(d), e = VALUES(e)""",(slno,a,b,c,d,e));
db.py:29: Warning: Incorrect integer value: '
' for column 'a' at row 1
  cursor.execute("""INSERT INTO dist VALUES('',%s,%s,%s,%s,%s,%s) ON DUPLICATE KEY UPDATE a = VALUES(a), b = VALUES(b), c = VA
LUES(c), d = VALUES(d), e = VALUES(e)""",(slno,a,b,c,d,e));
db.py:29: Warning: Data truncated for column 'b' at row 1
  cursor.execute("""INSERT INTO dist VALUES('',%s,%s,%s,%s,%s,%s) ON DUPLICATE KEY UPDATE a = VALUES(a), b = VALUES(b), c = VA
LUES(c), d = VALUES(d), e = VALUES(e)""",(slno,a,b,c,d,e));
db.py:29: Warning: Data truncated for column 'c' at row 1
  cursor.execute("""INSERT INTO dist VALUES('',%s,%s,%s,%s,%s,%s) ON DUPLICATE KEY UPDATE a = VALUES(a), b = VALUES(b), c = VA
LUES(c), d = VALUES(d), e = VALUES(e)""",(slno,a,b,c,d,e));
db.py:29: Warning: Data truncated for column 'd' at row 1
  cursor.execute("""INSERT INTO dist VALUES('',%s,%s,%s,%s,%s,%s) ON DUPLICATE KEY UPDATE a = VALUES(a), b = VALUES(b), c = VA
LUES(c), d = VALUES(d), e = VALUES(e)""",(slno,a,b,c,d,e));
db.py:29: Warning: Data truncated for column 'e' at row 1
  cursor.execute("""INSERT INTO dist VALUES('',%s,%s,%s,%s,%s,%s) ON DUPLICATE KEY UPDATE a = VALUES(a), b = VALUES(b), c = VA
LUES(c), d = VALUES(d), e = VALUES(e)""",(slno,a,b,c,d,e));
pi@raspberrypi:~/Desktop/dist $ 
```

Figure 22 Python Script after execution – db.py – Distance Matrix

The values are then inserted onto MySQL table. We then open ‘localhost/phpmyadmin’ page on web browser. We select the database and click on the table dist.


phpMyAdmin
Recent Favorites
New
Arithmetic
Distance
New
dist
log
information_schema
mysql
performance_schema
testdb
Server: localhost:3306 » Database: Distance » Table: dist
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code]
Show all Number of rows: 25 Filter rows: Search this table
+ Options
Check all With selected: Edit Copy Delete Export
Query results operations
Print Copy to clipboard Export Display chart Create view

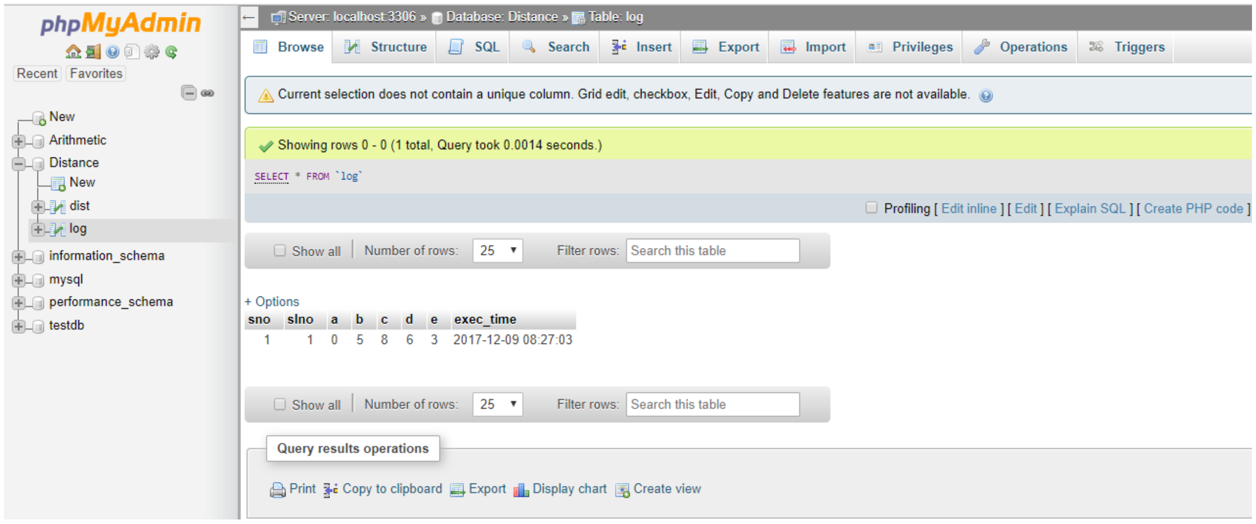
Showing rows 0 - 0 (1 total, Query took 0.0009 seconds.)

```
SELECT * FROM `dist`
```

	sno	slno	a	b	c	d	e
Edit Copy Delete	1	1	0	5	8	6	3

Figure 23dist table – PhpMyAdmin page

The values present here are also present in the ‘log’ table of the same database ‘Distance’. The values with time are stored here in the table.


phpMyAdmin
Recent Favorites
New
Arithmetic
Distance
New
dist
log
information_schema
mysql
performance_schema
testdb
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code]
Show all
+ Options
Query results operations
Print Copy to clipboard Export Display chart Create view

Server: localhost:3306 » Database: Distance » Table: log

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 0 (1 total, Query took 0.0014 seconds.)

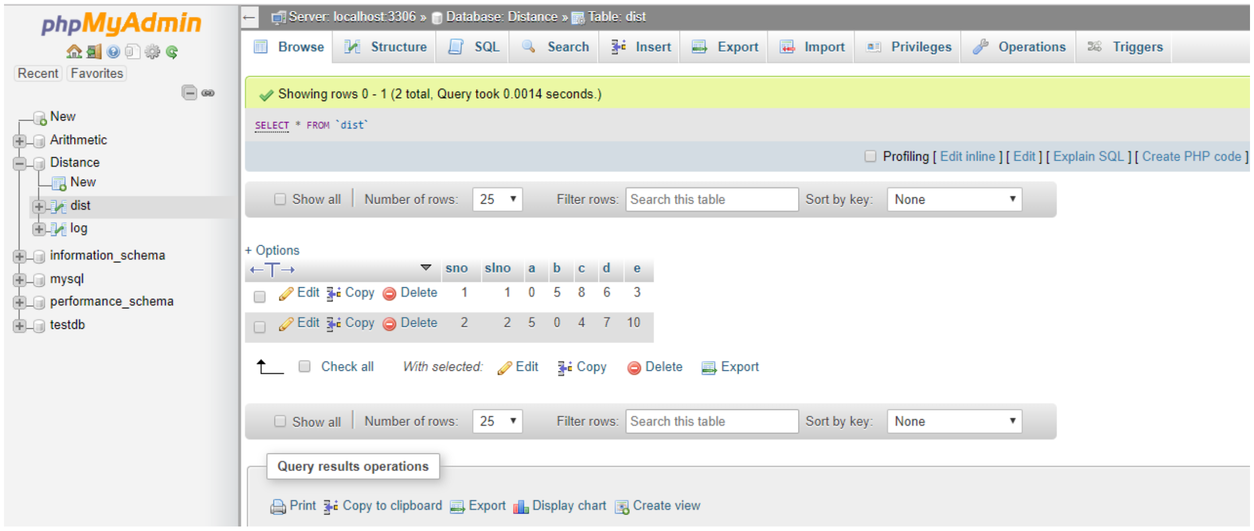
```
SELECT * FROM `log`
```

Number of rows: 25 Filter rows: Search this table

sno	slno	a	b	c	d	e	exec_time
1	1	0	5	8	6	3	2017-12-09 08:27:03

Figure 24log table – PhpMyAdmin page

Similarly, for slave 2, the process is the same and the inserted values are;


phpMyAdmin
Recent Favorites
New
Arithmetic
Distance
New
dist
log
information_schema
mysql
performance_schema
testdb
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code]
Show all
+ Options
Check all With selected: Edit Copy Delete Export
Query results operations
Print Copy to clipboard Export Display chart Create view

Server: localhost:3306 » Database: Distance » Table: dist

Showing rows 0 - 1 (2 total, Query took 0.0014 seconds.)

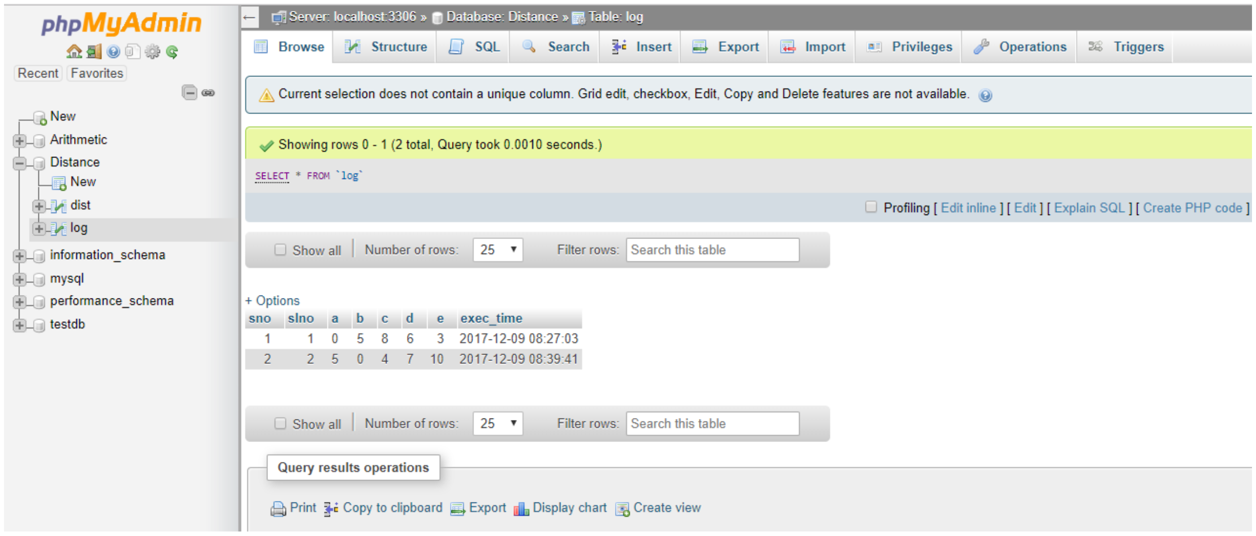
```
SELECT * FROM `dist`
```

Number of rows: 25 Filter rows: Search this table Sort by key: None

	sno	slno	a	b	c	d	e
Edit Copy Delete	1	1	0	5	8	6	3
Edit Copy Delete	2	2	5	0	4	7	10

Figure 25dist table for all slaves – PhpMyAdmin page

And the log table goes as;


phpMyAdmin
Recent Favorites
New
Arithmetic
Distance
New
dist
log
information_schema
mysql
performance_schema
testdb
Server: localhost:3306 » Database: Distance » Table: log
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Show all Number of rows: 25 Filter rows: Search this table
+ Options
Query results operations
Print Copy to clipboard Export Display chart Create view

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 1 (2 total, Query took 0.0010 seconds.)

```
SELECT * FROM `log`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code]

sno	slno	a	b	c	d	e	exec_time
1	1	0	5	8	6	3	2017-12-09 08:27:03
2	2	5	0	4	7	10	2017-12-09 08:39:41

Figure 26 log table for all slaves – PhpMyAdmin page

The final step is to transfer the ‘dist’ table to master Arduino. The master Arduino code needs to be edited for few lines since it will be reading Serial output. The script goes as:

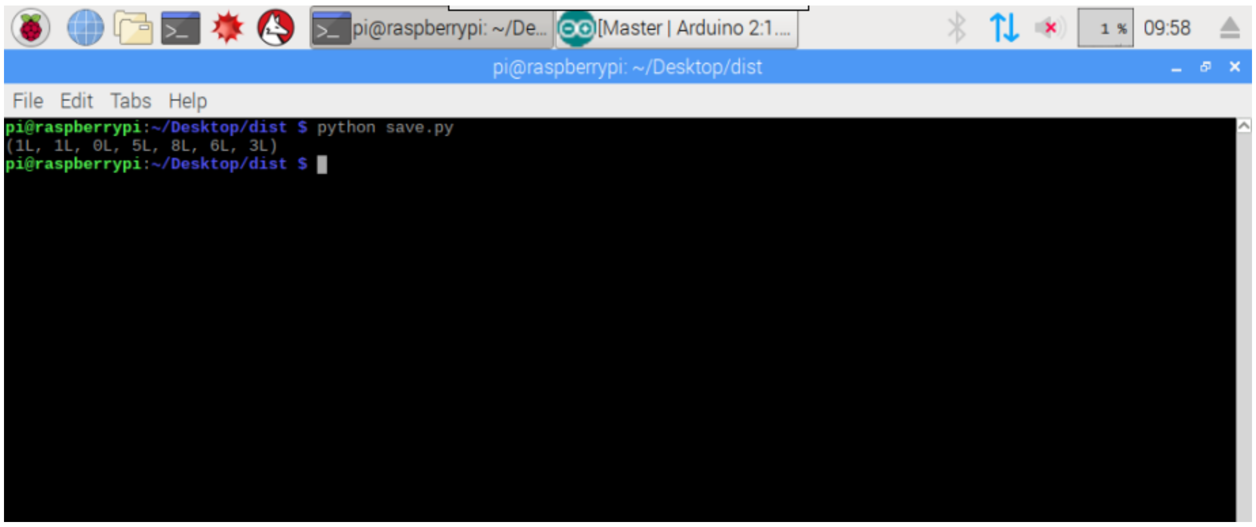
```
#include "RS485_protocol.h"
#include <SoftwareSerial.h>
const byte ENABLE_PIN = 2;
SoftwareSerial rs485 (10,11);// receive pin, transmit pin

char input[5];
int charsRead =0;
boolean received = false;
char res;
boolean sendnow=false;



void setup()
{

Serial.begin(9600);
Serial1.begin(9600);
  rs485.begin (28800);
  pinMode (ENABLE_PIN, OUTPUT);

}

void loop()
{
  Serial.flush();
if (Serial.available() == 0)
{

}

while(Serial.available() >0)
{
Serial.print(“ “);
Serial.print(Serial.read()); // this will read the inputs from MySQL table through python script
“save.py”
}

delay(200);
while (Serial.available() > 0)
{

charsRead = Serial.readBytesUntil('\n', input, sizeof(input)-1);
sendnow =true;
}
 if(sendnow)
 {
  digitalWrite (ENABLE_PIN, HIGH);  // enable sending
  rs485.write(input);
  Serial.println(input);
  delayMicroseconds (660);
  digitalWrite (ENABLE_PIN, LOW);
   sendnow = false;
 }

    delay(1000);
    digitalWrite (ENABLE_PIN, LOW);
   //Serial.println("Receiving Distance Matrix");
    while(rs485.available() && !sendnow)
    {
    int dist;
    //for(int i=0;i<5;i++){
    dist = rs485.read();
    digitalWrite (ENABLE_PIN, HIGH);
    //Serial.println(dist);
    //}
   }

}
```

Table 35 Master Arduino code for ‘dist’

From python script, if i=1; then only the first row will be retrieved and printed on the master serial output. The python script to be executed is:

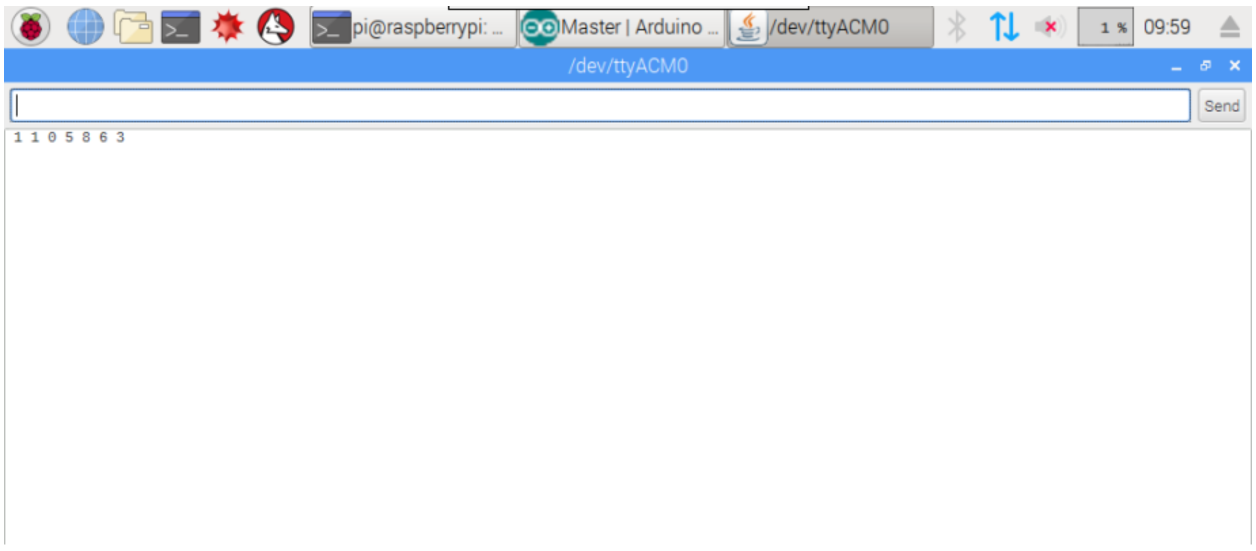
```
python save.py
```

Table 36 Python script executing – save.py



```
pi@raspberrypi:~/Desktop/dist $ python save.py
(1L, 1L, 0L, 5L, 8L, 6L, 3L)
pi@raspberrypi:~/Desktop/dist $
```

Figure 27 Python script for save.py – to Master



```
1 1 0 5 8 6 3
```

Figure 28 Retrieved values on Master Serial Output

The output will be printed onto the master serial output.

8.4 Arithmetic

In the similar way, we create a database called Arithmetic. We perform the same three steps as done under Distance Matrix. We create a table called 'calc':

```
create table calc (sno int(4) auto_increment unique not null, slno int(4) primary key not
null, input int(4) not null, result int(4) not null);
```

Table 37 Creating table calc for Arithmetic

We then create a table 'log' to store log values and create a trigger under 'calc' table as we had done in the 'Distance Matrix' task.

```
create table log (sno int(4) not null, slno int(4) not null, input int(4) not null, result
int(4) not null, exec_time datetime);
```

Table 38 Creating table log for Arithmetic

The python scripts logic will be the same for insert/update, however attributes will be changed, the script 'db.py' is;

```
import serial
import time
import MySQLdb as mdb

arduino = serial.Serial("/dev/ttyACM1")
arduino.baudrate=9600

data1 = arduino.readline()
data2 = arduino.readline()
data3 = arduino.readline()



slno = data1
input = data2
result = data3


con = mdb.connect('localhost', 'phpmyadmin', 'some_pass', 'Arithmetic');

with con:
   cursor = con.cursor()
   cursor.execute("""INSERT INTO calc VALUES("%s,%s,%s) ON DUPLICATE KEY
UPDATE input = VALUES(input), result = VALUES(result)""",(slno,input,result));
   con.commit()
   cursor.close()
```

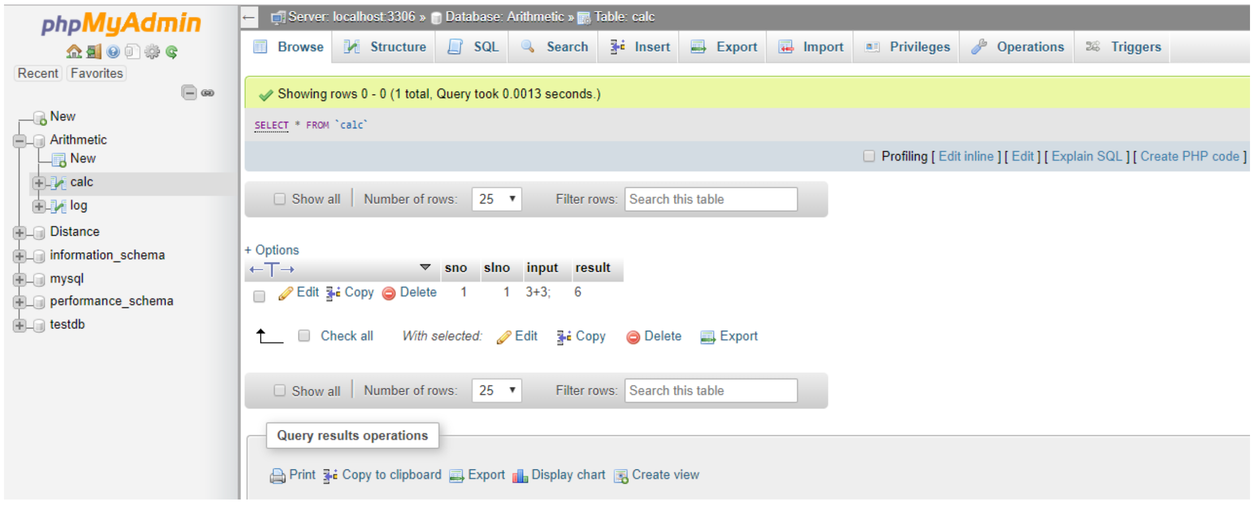
Table 39 Python script for Arithmetic – db.py

The similar way for 'save.py';

```
import MySQLdb as mdb
import serial

con = mdb.connect('localhost', 'phpmyadmin', 'some_pass', 'Arithmetic');

cur = con.cursor()

cur.execute("""select * from calc""")

row = cur.fetchall()

while row is not None:

    for i in range(len(row)):
        print row[i]
        ser = serial.Serial("/dev/ttyACM0")
        ser.baudrate=9600
        ser.write(row[i])

    break

cur.close()
```

Table 40 Python script for Arithmetic – save.py

The final step of executing everything, we get;

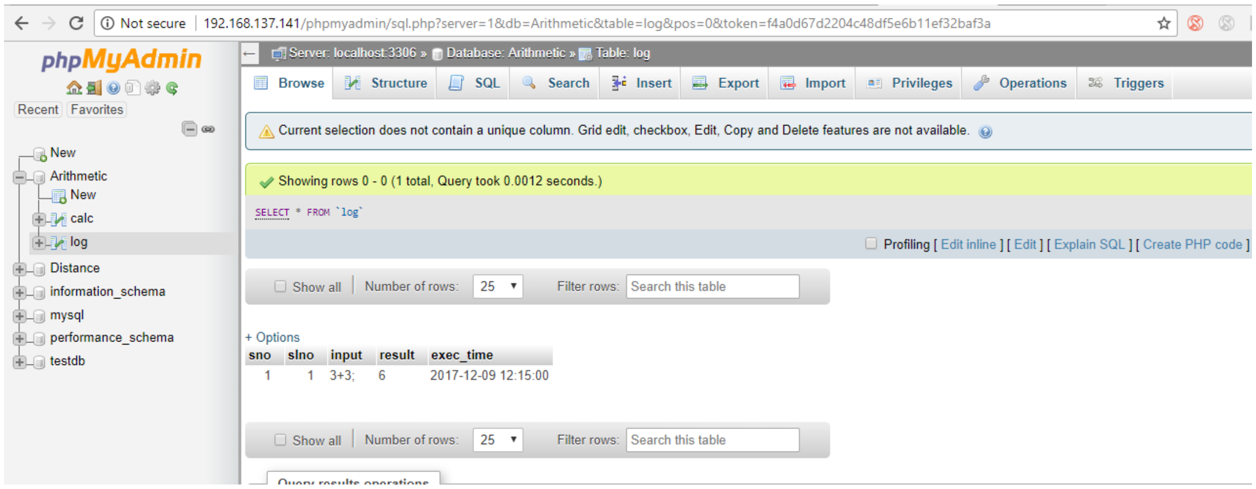

phpMyAdmin
Recent Favorites
New
Arithmetic
New
calc
log
Distance
information_schema
mysql
performance_schema
testdb
Server: localhost:3306 » Database: Arithmetic » Table: calc
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code]
Show all Number of rows: 25 Filter rows: Search this table
+ Options
Check all With selected: Edit Copy Delete Export
Query results operations
Print Copy to clipboard Export Display chart Create view

Showing rows 0 - 0 (1 total, Query took 0.0013 seconds.)

```
SELECT * FROM `calc`
```

	sno	slno	input	result
Edit Copy Delete	1	1	3+3;	6

Figure 29 calc table – PhpMyAdmin page


Not secure | 192.168.137.141/phpmyadmin/sql.php?server=1&db=Arithmetic&table=log&pos=0&token=f4a0d67d2204c48df5e6b11ef32baf3a
phpMyAdmin
Recent Favorites
New
Arithmetic
New
calc
log
Distance
information_schema
mysql
performance_schema
testdb
Server: localhost:3306 » Database: Arithmetic » Table: log
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code]
Show all | Number of rows: 25 Filter rows: Search this table
+ Options
Query results operations

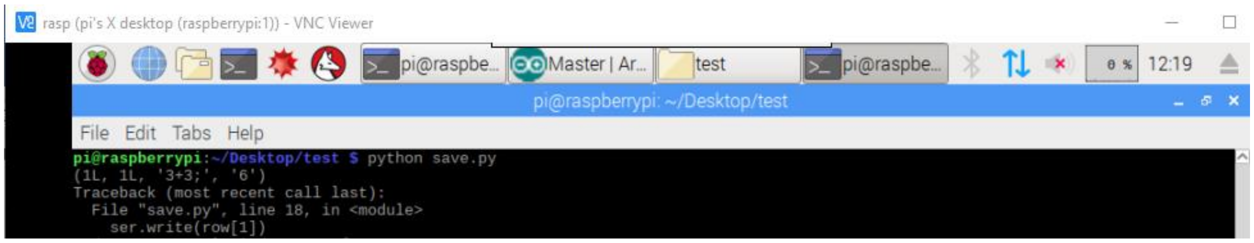
Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 0 (1 total, Query took 0.0012 seconds.)

```
SELECT * FROM `log`
```

sno	slno	input	result	exec_time
1	1	3+3;	6	2017-12-09 12:15:00

Figure 30 log table – PhpMyAdmin page


rasp (pi's X desktop (raspberrypi:1)) - VNC Viewer
pi@raspbe...
Master | Ar...
test
pi@raspbe...
12:19
pi@raspberrypi: ~/Desktop/test
File Edit Tabs Help

```
pi@raspberrypi:~/Desktop/test $ python save.py
(1L, 1L, '3+3;', '6')
Traceback (most recent call last):
  File "save.py", line 18, in <module>
    ser.write(row[1])
```

Figure 31 Python script save.py after execution

Chapter IX

9. Conclusion and Future Work

The given tasks on master to slave and slave to master communication was established successfully. With different objectives, we showed what the master and slaves are set to perform. Their functionalities when a request/respond message is sent/received. The ability to communicate was done faster and reliable through Mod Bus RS485.

Since it was a wired connection, there could be improvements in the objectives where ESP8266 Wi-Fi WLAN module can be used for wireless connections. Also, Bluetooth module HC-05 can be used for communication between different enabled-Bluetooth devices.

Another famous shield that is widely used is Xbee Shield. The module can communicate wirelessly up to 100-300 feet. Mostly used for broadcasting signals. It is used as a USB replacement which can be configured for mesh networking options.

REFERENCES

[1] **H, Mayer.** An Introduction to Conveyor Theory. Western Electric Engineer, 1960.

[2] **Qiming, Huang.** Analysis of Automated Picking system and its Application. 2002.

[3] **CEMA.** http://www.cemanet.org/. Conveyor Equipment Manufacturers Association.

[4] **Ji Yonghong and Zhu Zhonglong.** Automatic Passenger Transportation System at the West Terminal of Hongqiao International Airport. Urban Mass Transit, 2012.

[5] **ZHAO Jiong and XUE Jingsong.** Control Unit Based Path Finding and Scheduling System; Information and Control. 2003.

[6] **CHANG F, WANG P, QIAO Y.** Route Optimization and Dispatching for Transportation Net of Material Distributing system based on Intelligent and Algorithm. 2005.

[7] **Sergey Libert and Micheal ten Hompel.** Ontology-based Communication for the Decentralized material flow control of a conveyor facility, Logistic research. 2011.

[8] **Ying Wang.** High volume Conveyor Sortation System Analysis. 2006.

[9] **K.L. Mak and Peggy S.K Lau.** Order Pickings in an AS/RS with multiple IO stations using an Artificial Immune System with aging Antibodies. Engineering Letters, 2008.

[10] **Thomas Arzt and Freddy Bulcke.** A New Low Cost Approach in 200mm and 300mm AMHS. 2000.

[11] **W. Espelage and E. Wanke.** A Linear Time Approximation Algorithm for Movement Minimization in Conveyor Flowshop Processing. European Journal of Operational Research, 2005.

[12] **Hayslip, Nunzio.** A Reconfigurable Simulator for Coupled Conveyors. Thesis of the University of Akron, 2006.

[13] **Jack Purdum.** Beginning C for Arduino. Apress 1st ed, 2012.

[14] **Simon Monk.** Programming the Raspberry Pi: Getting Started with Python. Mcgraw-Hill Education Ltd, 2015.

[15] **Piotr J Kula.**Raspberry Pi 2 Server Essentials. Packt Publishing, 2016.

[16] **Adith Jagadish Boloor.**Arduino by Example. Packt Publishing, 2015.

[17] **Srihari Yamanoor and Sai Yamanoor.** Python Programming with Raspberry Pi. Packt Publishing, 2017.

[18] **Raspbian.** https://www.raspberrypi.org/downloads/raspbian/. Raspberry Pi Foundation.

[19] **Arduino Libraries.** https://www.arduino.cc/en/Reference/Libraries. Arduino

[20] **SoftwareSerial Library.** https://www.arduino.cc/en/Reference/SoftwareSerial. Arduino.

[21] **PuTTY.** https://www.chiark.greenend.org.uk/~sgtatham/putty. MIT License.

[22] **Advanced IP Scanner.** http://www.advanced-ip-scanner.com/. Famatech.

[23] **VNC Viewer.** https://www.realvnc.com/en/connect/download/viewer/. RealVNC.

[24] **Nick Gammon.** RS485 Library. https://www.gammon.com.au/forum/?id=11428. Creative Commons Attribution 3.0 Australia License.

[25] **Modelio.** Sequence Diagrams. https://www.modelio.org/. GLP License and Apache Public License(APL).

[26] **Arduino Version 1.8.5.** https://www.arduino.cc/en/Main/Software. Arduino.