TRANSMISI : JURNAL ILMIAH TEKNIK ELEKTRO, 25, (3), JULI 2023
p-ISSN 1411-0814 e-ISSN 2407-6422

A CONTROLLER AREA NETWORK (CAN) BUS TEMPERATURE AND HUMIDITY DATA MONITORING SYSTEM

Arief Wisnu Wardhana[*)], Agung Mubyarto and Acep Taryana

Department of Electrical Engineering, Faculty of Engineering, Universitas Jenderal Soedirman, Purwokerto
Jl. Mayjend. Sungkono Km 5, Blater, Kalimanah, Purbalingga 53371, Indonesia

[*)] *E-mail: arief.wardhana@unsoed.ac.id*

Abstrak

Pengukuran suhu dan kelembapan sering diperlukan untuk banyak bidang, misalnya pertanian, industri, atau penelitian. Artikel ini memaparkan riset tentang sistem pemonitor suhu dan kelembapan relatif yang mampu mencatat data secara otomatis dan terus-menerus sepanjang hari. Jaringan sistem ini berbasis standar bus CAN (Controller Area Network) yang didesain agar memungkinkan banyak node kendali untuk berkomunikasi satu sama lain tanpa sebuah komputer host. Sistem terdiri dari sebuah jalur bus CAN, dua node transmisi yang dijalankan oleh Arduino UNO dan Arduino Nano, serta satu node penerima Arduino UNO board. Kemudian terdapat tiga MCP2515 CAN bus controllers, tiga TJA1050 CAN transceivers, dua sensor suhu dan kelembapan DHT11, dan sebuah LCD I2C 16x2 untuk menampilkan data suhu dan kelembapan. Terdapat dua node transmisi, satu mengirimkan data suhu dan satunya lagi mengirimkan data kelembapan. Data kemudian diproses oleh node transmitter dan dikirimkan melalui CAN bus. Untuk menampilkan data dilakukan oleh node receiver. Pengujian transmisi dari beberapa nomor identifikasi pesan yang berbeda menunjukkan bahwa LCD selalu menampilkan pesan yang mempunyai nomor identifikasi lebih rendah. Dengan modifikasi program untuk node transmitter, data suhu dan kelembapan terukur ditampilkan secara bergantian dan kontinyu pada LCD. Secara keseluruhan, sistem ini sudah berfungsi dengan baik sesuai dengan spesifikasi awal. CAN bus bisa mentransfer sampai delapan byte dalam sekali waktu. Terutama, skema alokasi prioritas pesan pada identifier adalah satu fitur CAN yang membuatnya sangat menarik untuk digunakan pada lingkungan kendali waktu-nyata. Selain itu, sistem ini sudah terbukti stabil dan dapat diandalkan.

Kata kunci: CAN bus, mikrokontroler, sensor suhu, arbitrasi, nomor identifikasi pesan, jaringan bus perangkat

Abstract

Measurement of temperature and humidity is required in many fields, e.g. agriculture, industrial, or research. This paper presents research about a temperature and relative humidity monitoring device automatically and continuously for 24 hours daily. The network is based on the Controller Area Network (CAN) bus standard, which allow multiple controller nodes to communicate with each other without a host computer. The system consists of a CAN bus line, two transmitter nodes by Arduino UNO board and Arduino Nano board, one receiver node by one Arduino UNO board, three MCP2515 CAN bus controllers, three TJA1050 CAN transceivers, two DHT11 temperature and relative humidity sensors, and an I2C LCD 16x2 for data display. There are two transmitter nodes for temperature and humidity data. The data is then processed by the transmitter nodes and sent via CAN bus. Display of data is carried out by the receiver. Transmission tests of different message ID both for temperature and humidity messages show that the LCD always displays the messages that have lower ID. By modifying the programs, measured data can be displayed alternately and continuously on the LCD. Overall, the system functions according to initial specifications. CAN bus network can transfer up to eight bytes of information at a time. Especially, the allocation of priority to messages in the identifier is a feature of CAN that makes it particularly attractive for use within a real-time control environment. Furthermore, the system has been proved stable and reliable.

Keywords: CAN bus, microcontroller, temperature sensor, arbitration, message identification number, device bus network

1. Introduction

Today's science and technology has developed very rapidly, especially in things that can help human life and work so that it becomes easier and more efficient. Humans have developed so much from using primitive methods to reach today's modern conditions in the way they carry out their daily lives. Some examples showing human activities in their in daily life, are in using various tools to help them,

processing food ingredients for their food needs, or even just to feel and observe the weather in their vicinity.

For example, a person might just feel and observe the temperature or humidity around them without any further purpose. This activity can also be carried out even further, namely to deliberately monitor the temperature and humidity continuously for certain purposes. This certainly requires an accurate recording device.

If we are to measure temperature or humidity, we must have some sort of units by which to classify these measurements. The original units used were 'hot' and 'cold' [1]. And only the terms 'moist' and 'dry' were used to classify humidity measurements. This may have been sufficient for their time, but are inadequate for modern use. In addition, due to the nature of the measurement, it is sometimes required for a continuous monitoring. For example, we often need to record (or to monitor) the temperature or humidity that changes every minute or even seconds. If done manually, this will become inefficient because it takes a lot of time and makes it difficult to collect data. Therefore, an easy- to-use data monitoring system is needed.

Various types of temperature and humidity data monitoring system could be designed. It might be in the form of a stand-alone temperature and humidity data monitoring consisting of a single device. On the other hand, it could be a collection of several such device connected in a network. For the later case, a method (called a standard or protocol) is certainly required to regulate the data transmission between the devices. Several protocols have been standardized, that have even been included in the industrial network such as MODBUS standard, Profibus standard, CAN bus standard.

A lot of research about temperature and humidity monitoring has been done. Some of them consists of several sensor devices connected in a network. Those which are designed based on standard CAN bus can mentioned as follows.

First, there is research done by J. da Silva Sa et al presented the implementation of smart sensors for monitoring temperature and for communicating among themself using the protocol CAN [2]. Next, a work done by Xu Yan et al introduced data acquisition system design composed of temperature and humidity sensor, the SCM system, computer, the CAN bus [3]. There is also another research performed by Q. Zhu et al proposed a temperature remote monitoring embedded system platform. The embedded microprocessor AT91SAM7X256 is used as CPU of the system. The system realizes real-time remote data collection monitoring and storage through protocol data conversion of the CAN bus and RS232 bus of the distributed temperature acquisition node [4]. And research aiming at the monitoring environment characteristics, an on-line monitoring system of power cable joints temperature based on the combination of CAN wired transmission and ZigBee wireless network is designed, done by Lihong Zhang et al [5].

The network of temperature and humidity monitoring devices that was designed in this research was also based on the CAN bus standard, which is the CAN2.0b Standard. The system consists of three CAN nodes with their respective CAN controllers and CAN transceivers. Two nodes will do the transmission of temperature and humidity data through the bus, and another one node will carry out displaying the data. A standard CAN with 11-bit identifier will be used as the data CAN frame format.

This device is able to monitor temperature and humidity in the vicinity of sensor location. The temperature and humidity sensors can be placed in several different locations. In this experiment, two humidity and temperature sensors were installed in two different locations. However, this is an open system, in the sense that it is possible to have up to 8 device nodes on a single CAN bus since on a standard CAN with 11-bit identifier, up to 8 bytes data may be transmitted. In other words, addition of some more measurement points is possible. An example of object whose temperature and relative humidity can be measured by this device is mainly a room that is sensitive to changes in temperature and relative humidity.

For maximum cable and maximum number of nodes for this CAN bus, the High-Speed ISO 11898 Standard specifications are given for a maximum signaling rate of 1 Mbps with a bus length of 40 m and a maximum of 30 nodes.

2. Controller Area Network (CAN) Bus

CAN bus is a device bus network based on the widely used CAN electronic chip technology. It was originally used inside automobiles to control internal components such as brakes and other systems [6]. The CAN bus was developed by BOSCH as a multi-master, message broadcast system that specifies a maximum signaling rate of 1 megabit per second (bps) [6]. Unlike a traditional network such as USB or Ethernet, CAN does not send large blocks of data point-to-point from node A to node B under the supervision of a central bus master. In a CAN network, many short messages (up to eight bytes) like temperature, humidity, or RPM are broadcast to the entire network, which provides for data consistency in every node of the system.

2.1. Standard CAN

The CAN communication protocol is a carrier-sense, multiple-access protocol with collision detection and arbitration on message priority (CSMA/CD+AMP). CSMA means that each node on a bus must wait for a prescribed period of inactivity before attempting to send a

message. CD+AMP means that collisions are resolved through a bit-wise arbitration, based on a preprogrammed priority of each message in the identifier field of a message. The higher priority identifier always wins bus access. That is, the last logic-high (in CAN, a logic-high is associated with a zero) in the identifier keeps on transmitting because it is the highest priority. Since every node on a bus takes part in writing every bit "as it is being written," an arbitrating node knows if it placed the logic-high bit on the bus [7].

The ISO-11898:2003 Standard CAN, with the standard 11-bit identifier, provides for signaling rates from 125 kbps to 1 Mbps.The bit fields of Standard CAN are shown in Figure 1 below [7].

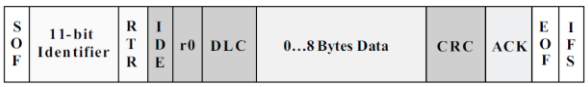

Figure 1. Standard CAN: 11-bit Identifier

The Standard CAN 11-bit identifier establishes the priority of the message. The lower the binary value, the higher its priority. Data – up to $8\ x\ 8\ bytes\ = 64$ bits of application data may be transmitted.

2.2. A CAN Message

The CAN bus access is regulated by the method of nondestructive bitwise arbitration. Nondestructive means that the frame that is winner of the arbitration i.e. the higher priority message, is not disturbed and does not need to be restarted. This mechanism requires the relevant physical drivers to be implemented in a certain way: The two logical levels on the CAN bus must be dominant and recessive, meaning that one node, sending a dominant level, overwrites all other nodes that send a recessive level. In the CAN protocol, a logical one is sent recessive and a logical zero is sent dominant.

A fundamental CAN characteristic shown in Figure 2 is the opposite logic state between the bus, and the driver input and receiver output. Normally, a logic-high is associated with a one, and a logic-low is associated with a zero - but not so on a CAN bus [7].

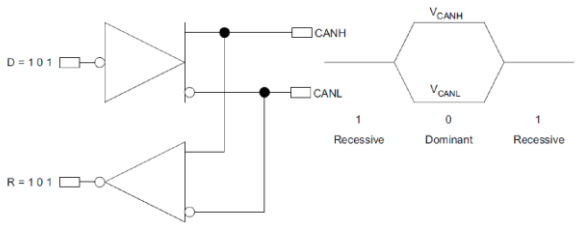

Figure 2. The Inverted Logic of a CAN Bus

Bus access is event-driven and takes place randomly. If two nodes try to occupy the bus simultaneously, access is implemented with a non-destructive, bit-wise arbitration. Non-destructive means that the node winning arbitration just continues on with the message, without the message being destroyed or corrupted by another node. The lower the binary message identifier number, the higher its priority. An identifier consisting entirely of zeros is the highest priority message on a network because it holds the bus dominant the longest. Therefore, if two nodes begin to transmit simultaneously, the node that sends a last identifier bit as a zero (dominant) while the other nodes send a one (recessive) retains control of the CAN bus and goes on to complete its message. A dominant bit always overwrites a recessive bit on a CAN bus [7].

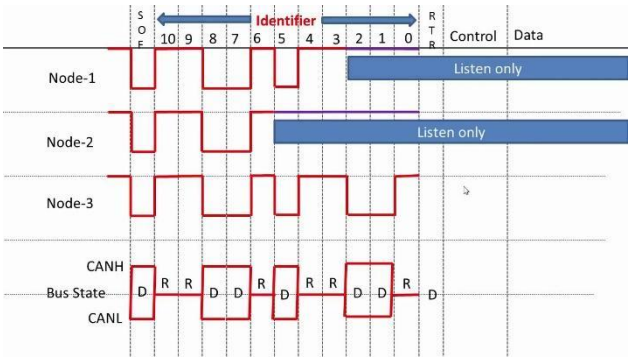

Figure 3. CAN Arbitration

An example below illustrates arbitration between three competing CAN nodes (see Figure 3). The figure displays the CAN arbitration process that is handled automatically by a CAN controller.

Three CAN nodes, node-1, node-2, and node-3 start a transmission at the same time. It is shown that the 11-bit message identifier for node-1 is 110 0101 1101 ($Id\ = 0 \times 65D$), message identifier for node-2 is 110 0111 0110 ($Id\ = 0 \times 676$), and message identifier for node-3 is 110 0101 1001 ($Id\ = 0 \times 659$).

According to the Carrier Sense Multiple Access with Collision Detection and Arbitration on Message Priority (CSMA/CD + AMP) access method, those three nodes had to wait until the bus is free (Carrier Sense). When this is detected, all of them send their *dominant Start Of Frame (SOF)* bit (Multiple Access). Note that a transmitting node constantly monitors each bit of its own transmission.

Throughout the frame, each CAN node will, via its transceiver, read back the logical value which occurs on the CAN bus and compare it with the transmitted logical value (Collision Detection). The three *dominant Start Of Frame* bits are superimposed on the bus and a *dominant* bus level is read back by all those three nodes. Next, the MSB of the identifier is sent. In the example, the MSB is *recessive* in all three identifiers. Here, too, the *recessive* level appears

on the bus and is again recognized by all three nodes. Therefore, no node notices the competing node (shown up to identifier bit 6) until the first difference in the identifier [8].

In the example, the first difference between two frames is at identifier bit 5. Node-1 and node-3 send a *dominant* level, node-2 sends a *recessive* level. According to the protocol specification, a *dominant* level appears on the bus. Node-1 and node-3 reads back the level that it had sent; hence, it does not see a collision. Node-2 also reads back the *dominant* level and compares it with the *recessive* level that it had sent; therefore, it sees a *Bit-Error*. At this point, node-2 recognizes that it has lost arbitration (Arbitration on Message Priority) and immediately stops the transmission of its own frame [8].

Next difference between two frames is at identifier bit 2. Node-3 send a *dominant* level, node-1 sends a *recessive* level. According to the protocol specification, a *dominant* level appears on the bus. Node-3 reads back the level that it had sent; hence, it does not see a collision. Node-1 also reads back the *dominant* level and compares it with the *recessive* level that it had sent; therefore, it sees a *Bit-Error*. At this point, node-1 recognizes that it has lost arbitration and immediately stops the transmission of its own frame [8].

Furthermore, now node-1 and node-2 become receiver of node-3's frame, because the frame that has won the arbitration may contain data that need to be processed by node-1 and node-2 [8].

Thus, in those example, node-3 has won arbitration. This is because node-3 has the lowest the binary message identifier number ($Id = 0 \times 659 = 110\ 0101\ 1001\ is\ the\ lowest\ identifier\ number$).

Note that node-3 is the node that sends a last identifier bit as a zero (dominant). See the difference between the four LSBs 1101 for node-1 and the four LSBs 1001 for node-3. Therefore, this node-3 will just continue on with the message, without the message being destroyed or corrupted by another node.

2.3. Bus Termination

Electrical signals on the bus are reflected at the ends of the electrical line unless preventive actions are taken. Signal reflection occurs when a signal is transmitted along a transmission medium, such as a copper cable or an optical fiber. Some of the signal power may be reflected back to its origin rather than being carried all the way along the cable to the far end. This happens because imperfections in the cable cause impedance mismatches and non-linear changes in the cable characteristics. These abrupt changes cause some of the transmitted signal to be reflected. To avoid mismatches when a node reads the bus electrical status, signal reflections must be avoided. Terminating the bus line with a termination resistor at both ends of the bus and avoiding unnecessarily long stubs lines of the bus is the best corrective action. The largest possible simultaneous transmission rate and bus length line are achieved by using a structure as close as possible to single line structure and by terminating both ends of the line (this layout is also referred to as *linear*). Specific recommendations for this structure can be found in the related standards (i.e. ISO11898-2 and -3). The method of terminating CAN hardware varies depending on the physical layer of the hardware itself (high-speed, low-speed, single-wire, or software-selectable). For high-speed CAN, both ends of the pair of signal wires (CAN H and CAN L) must be terminated. The termination resistors on the cable should match the nominal impedance of the cable [9].

ISO 11898 requires a cable with a nominal impedance of 120 Ω, and therefore 120 Ω resistors should be used for termination. If multiple devices are placed along the cable, only the devices on the ends of the cable need termination resistors [9].

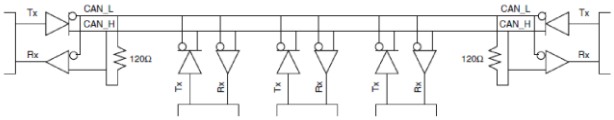

Figure 4. Terminating a High-Speed Network.

Figure 4 above gives an example of how to terminate a high-speed network. The termination can be of basic type (a single resistor as in the figure) or of split type (two 60 Ω resistors with an intermediate capacitor between 10 nF and 100 nF connected to ground) [9].

2.4. Details of a CAN Bus

The data link and physical signalling layers, which are normally transparent to a system operator, are included in any controller (embedded or stand-alone) that implements the CAN protocol. Connection to the physical medium is then implemented through a line transceiver to form a system node as shown in Figure 5 [7].

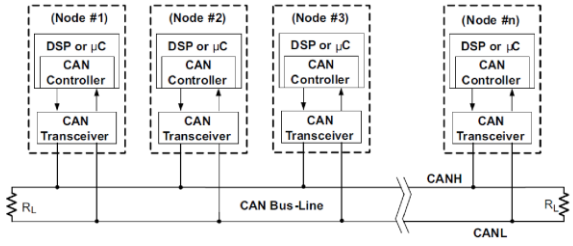

Figure 5. Details of a CAN Bus

3. The System Overview

The temperature and relative humidity monitoring system designed here will consist of a CAN bus interface which consists of a CAN bus line, several CAN nodes, several CAN controllers, several CAN transceivers, several sensors, one display device, and two terminating resistors.

The CAN bus line will be of the form two lines bus. One line is the CAN bus High line and the other is the CAN bus Low line.

The first node is a receiver node. This node will be carried out by an Arduino UNO R3. This board is the perfect board to get familiar with electronics and coding. This versatile microcontroller is equipped with the well-known ATmega328P and the ATMega 16U2 Processor [10]. Next node is a transmission node for temperature data. This will also be carried out by an Arduino UNO board. And the third node is a transmission node for humidity data. This humidity node will be carried out by an Arduino® Nano. This board is an intelligent development board designed for building faster prototypes with the smallest dimension. At the heart of the board is ATmega328 microcontroller clocked at a frequency of 16 MHz The board offers 22 digital input/output pins, 8 analog pins, and a mini-USB port [11].

For the CAN controller, the MCP2515 chip CAN controllers are used. Microchip Technology's MCP2515 is a stand-alone Controller Area Network (CAN) controller that implements the CAN specification, Version 2.0B. It is capable of transmitting and receiving both standard and extended data and remote frames. The MCP2515 has two acceptance masks and six acceptance filters that are used to filter out unwanted messages, thereby reducing the host MCU's overhead. The MCP2515 interfaces with microcontrollers (MCUs) via an industry standard Serial Peripheral Interface (SPI) [12].

Next is the CAN transceiver. The transceiver transmits and receives the physical data to and from the bus. TJA1050 chip transceivers are used here. The TJA1050 is the interface between the CAN controller and the physical CAN bus. The device provides differential transmit capability to the bus and differential receive capability to the CAN controller. It is fully compatible to the "ISO 11898" standard [13].

Depicted in a diagram, CAN bus interface of the system will look like Figure 6.

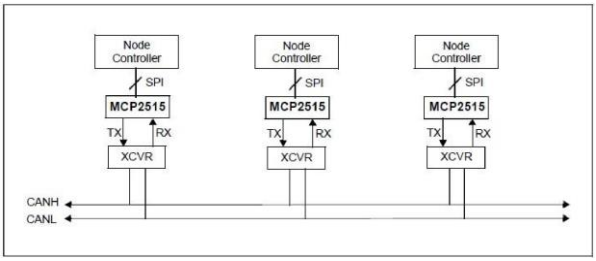

Figure 6. CAN Bus Interface of the Temperature and Humidity Monitoring System

In Figure 6 above, the blocks indicated by node controllers are the Arduino boards with their temperature & humidity sensors and LCD display. While the blocks indicated by XCVR are the TJA1050 CAN transceivers.

As mentioned above, the system will also have some means for communicating with outside world. These are sensors and a display device. For sensors, DHT11 humidity and temperature sensors are used. DHT11 temperature & humidity sensor features a temperature & humidity sensor complex with a calibrated digital signal output. By using the exclusive digital-signal-acquisition technique and temperature & humidity sensing technology, it ensures high reliability and excellent long-term stability. This sensor includes a resistive-type humidity measurement component and an NTC temperature measurement component, and connects to a high-performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost-effectiveness [14].

For the display device, a 16×2 LCD with an I2C interface is used. This is I2C interface 16x2 LCD display module, a high-quality 2 line 16 character LCD module with on-board contrast control adjustment, backlight and I2C communication interface

4. Method

The method used in this research includes programming the three nodes and assembling the various system's parts.

For the programming, all the Arduino boards will be programmed with Arduino IDE. Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. We can tell our board what to do by sending a set of instructions to the microcontroller on the board. To do so, we use the Arduino programming language (based on wiring), and the Arduino Software (IDE), based on processing. Each board will be programmed according to its function. Thus, in here there will be three different Arduino programs.

For the hardware, all the parts of the system will be assembled according to diagram in Figure 5 and Figure 6. Thus, there are three readily programmed nodes will be connected to the CAN bus line. Two nodes at the ends of the line are the receiver node and the humidity node. The temperature node is placed in the middle of the line, tapping the line. There also will be two resistors act as termination line resistors at the ends of the line.

Furthermore, we will have an experiment with the 11-bit message identifier number. That is, the pair of message identifier number (one for temperature message and the other for humidity message) was varied several times. For each pair, an experiment transmitting temperature dan humidity data was done. The results were then observed on the display device.

5. Results and Discussion

The results will also consist of two parts, the programming results and the hardware results. The programming results consists of three Arduino programs for the three nodes. One program for each node. While hardware results will be the assembled parts and the results from the LCD display.

5.1. Programming Results

There are three Arduino programs that has been written for each node. First is the program listing for the receiver node, whose flowchart is shown in Figure 7.

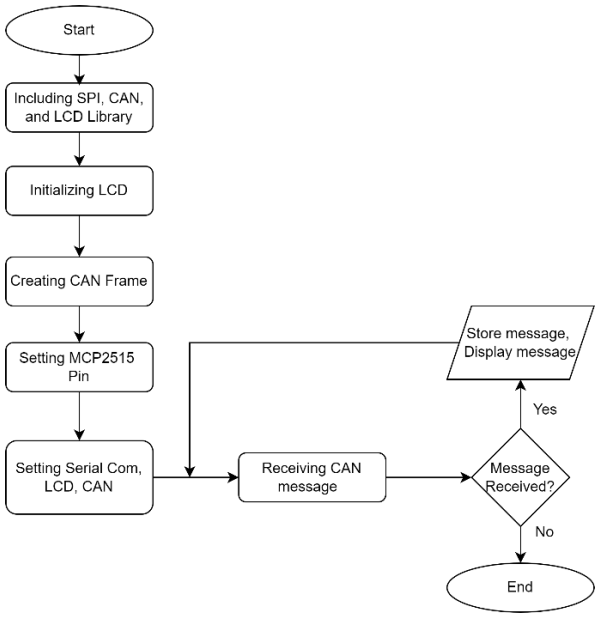

Figure 7. Flowchart for Receiver Node Program

From the flow chart above, the Arduino program listing for the receiver was created as shown in Figure 8. An explanation for each line is provided on the right.

It is shown that the receiver node has been programmed in such a way that this node will only be able to receive and display humidity data sent by humidity node and also receive and display temperature data sent by temperature node. This receiving node does not transmit data.

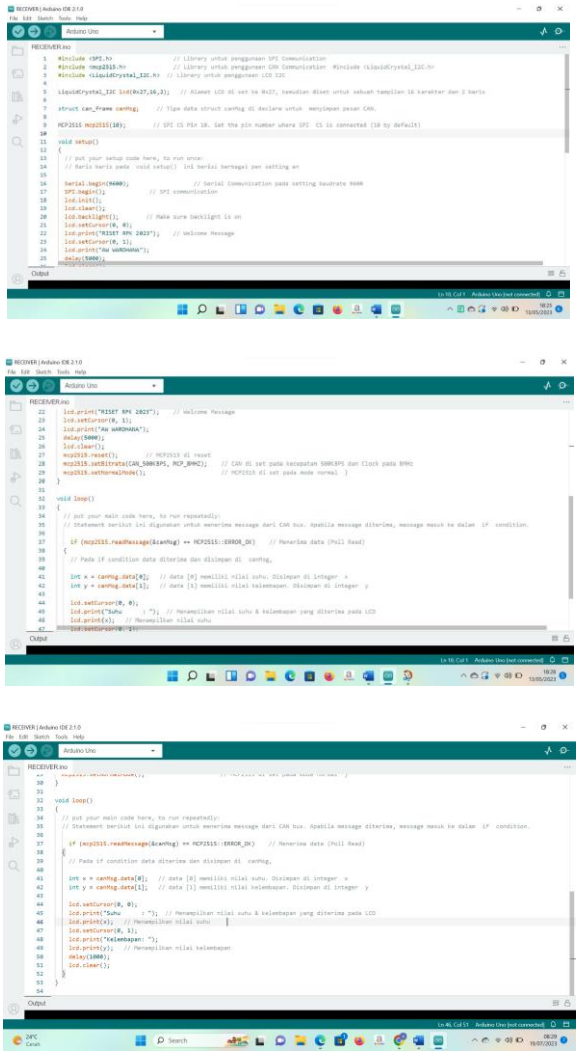

Figure 8. Arduino Program for Receiving Node

Some libraries are used for implementing SPI communication, implementing CAN communication, and using the LCD I2C .

After including libraries, LCD setting was done, and a structure for storing CAN message was declared.

Inside `void setup()` is various settings for SPI, LCD, and CAN. Inside `void loop()`, a statement for reading the CAN messages sent by transmitters (CAN messages contain temperature and humidity data) was created, and integers were also created for storing the temperature and humidity data. And some statements associated with the LCD for displaying the data.

Next is the the program listing for the node transmitting temperature data. The flowchart for this program is shown in Figure 9.

From those flow chart, the Arduino program listing for this temperature node was created as shown in Figure 10. An explanation for each line is again provided on the right.

This temperature node has been programmed in such a way that it will be able to transmit only temperature data recorded by the DHT11 sensor.

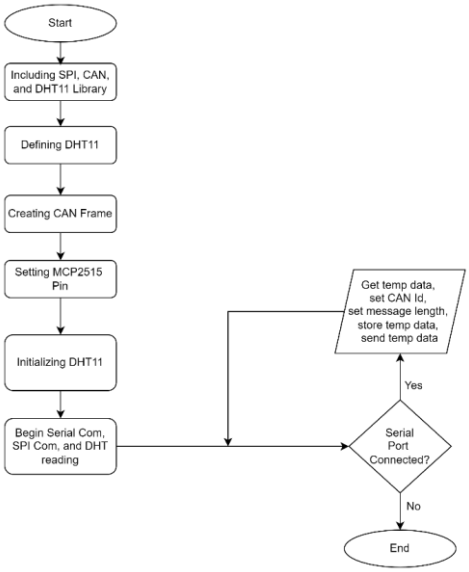

Figure 9. Flowchart for Temperature Node Program

Some libraries are used for implementing SPI communication, implementing CAN communication, and using the DHT11 sensor.

Inside `void setup()` is various settings for SPI, DHT11, and CAN. Inside `void loop()`, a statement for reading temperature and storing it in an integer was created. Then, it is shown that an example of message identifier for temperature which is $0x036$ ($Id = 000\ 0011\ 0110$). It is also shown that CAN message length is 8 bytes, shown as `canMsg.data[0]`,`canMsg.data[1]`,`canMsg.data[2]`,`canMsg.data[3]`,`canMsg.data[4]`,`canMsg.data[5]`,`canMsg.data[6]`, and `canMsg.data[7]`. Each is one byte long. This is in accordance with the structure of the standard CAN which has 0 … 8 btyes data. The temperature data is put on `canMsg.data[0]`. The other bytes are filled with $0x00$. The `void loop()` ends with a statement associated with MCP2515 for sending the CAN message.

And finally, is the the program listing for the node transmitting humidity data. The flowchart for this program is shown in Figure 11.

From those flow chart, the Arduino program listing for this temperature node was created as shown in Figure 12. An explanation for each line is provided on the right.

This humidity node has been programmed in such a way that it will be able to transmit only humidity data recorded by the DHT11 sensor.

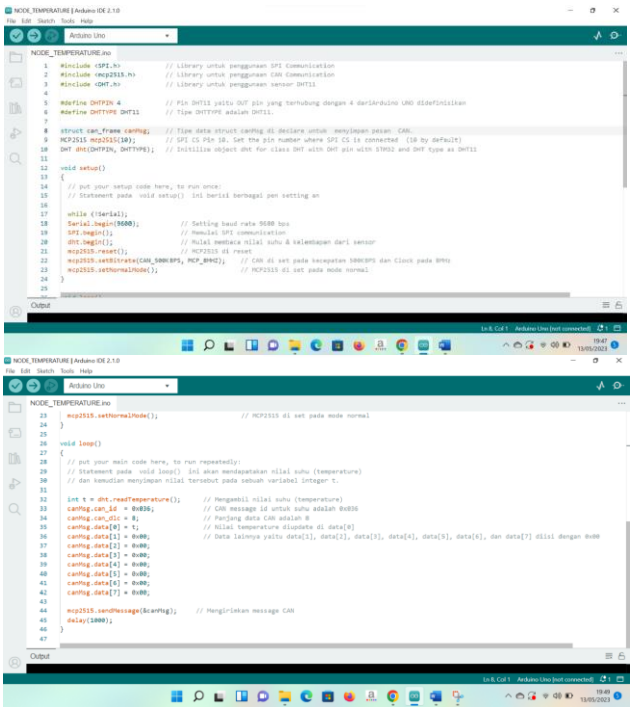

Figure 10. Arduino Program for Temperature Node

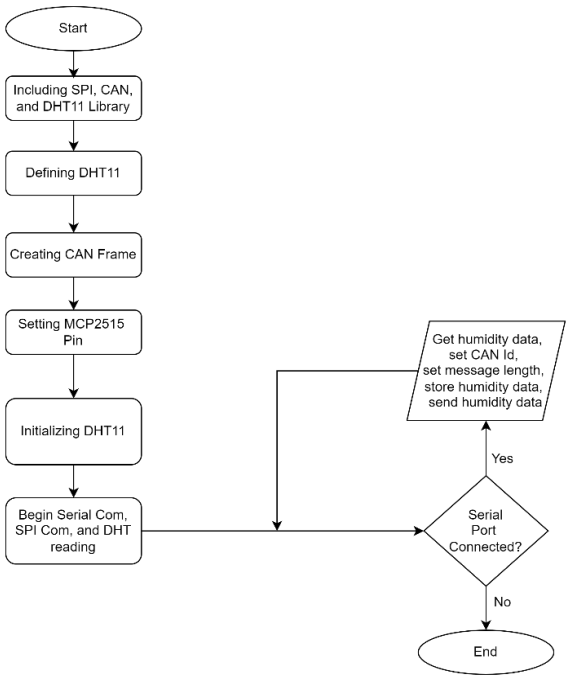

Figure 11. Flowchart for Humidity Node Program

Some libraries are used for implementing SPI communication, for implementing CAN communication, and for using the DHT11 sensor.

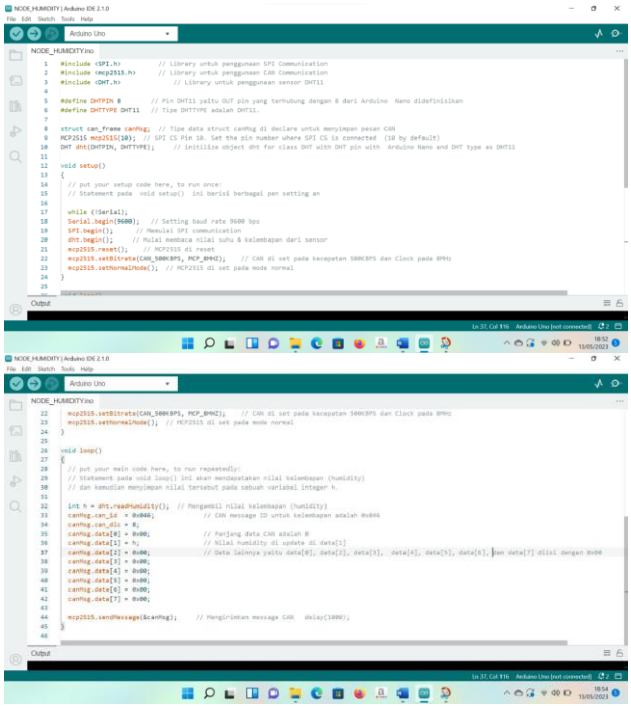

Figure 12. Arduino Program for Humidity Node

Inside `void setup()` is various settings for SPI, DHT11, and CAN. Inside `void loop()`, a statement for reading temperature and storing it in an integer was created. Then, it is shown an example of message identifier for humidity which is $0x046$ ($Id = 000\ 0100\ 0110$). It is also shown that CAN message length is 8 bytes, shown as `canMsg.data[0]`,`canMsg.data[1]`,`canMsg.data[2]`,`canMsg.data[3]`,`canMsg.data[4]`,`canMsg.data[5]`,`canMsg.data[6]`, and `canMsg.data[7]`. Each is one byte long. This is in accordance with the structure of the standard CAN which has 0 … 8 btyes data. The temperature data is put on `canMsg.data[1]`. The other bytes are filled with $0x00$. The `void loop()` again ends with a statement associated with MCP2515 for sending the CAN message.

5.2. Hardware Results

The assembled parts are shown in Figure 13. They are the temperature node, the humidity node, the receiver node, two DHT11 temperature & humidity sensors, an LCD, three MCP2515 & TJA1050 modules (one module for each node), and two bus termination resistors of magnitude 120 Ω.

Two nodes at the end of the CAN bus line are the temperature node and the receiver node. The humidity node is placed in the middle of the line, tapping the line. The two termination 120 Ω resistors shows that this system employs a parallel termination scheme. Parallel termination is one of the most prevalent termination schemes today. Parallel termination employs a resistor across the differential lines at the far (receiver) end of the transmission line to eliminate all reflections [15].

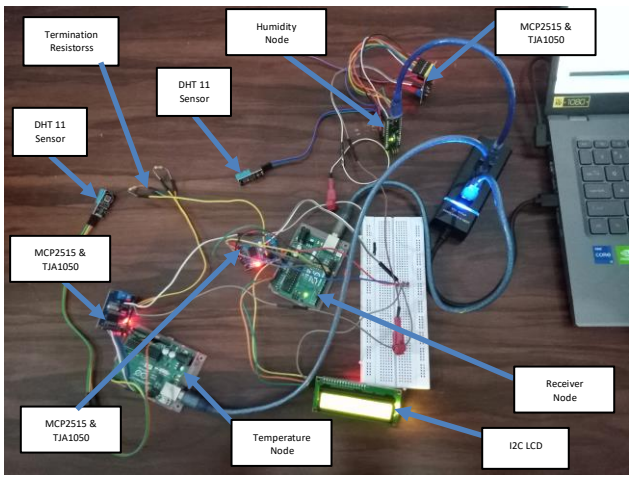

Figure 13. Assembled of the Three Nodes

Table 1 below shows the results when the pair of message identifier number (one id for temperature message and the other id for humidity message) was varied several times.

Table 1. LCD Results for Various Pair of Message Identification Number

LCD Display Results	Humidity Node Message Id Number	Temperature Node Message Id Number
Hum (82%)	$Id = 0 \times 001$	$Id = 0 \times 0FF$
Hum (82%)	$Id = 0 \times 011$	$Id = 0 \times 0EE$
Hum (82%)	$Id = 0 \times 022$	$Id = 0 \times 0DD$
Hum (82%)	$Id = 0 \times 033$	$Id = 0 \times 0CC$
Hum (82%)	$Id = 0 \times 044$	$Id = 0 \times 0BB$
Hum (82%)	$Id = 0 \times 055$	$Id = 0 \times 0AA$
Hum (82%)	$Id = 0 \times 066$	$Id = 0 \times 099$
Hum (82%)	$Id = 0 \times 077$	$Id = 0 \times 088$
Hum (82%)	$Id = 0 \times 043$	$Id = 0 \times 046$
Temp(28°C)	$Id = 0 \times 039$	$Id = 0 \times 036$
Temp(28°C)	$Id = 0 \times 088$	$Id = 0 \times 077$
Temp(28°C)	$Id = 0 \times 099$	$Id = 0 \times 077$
Temp(28°C)	$Id = 0 \times 0AA$	$Id = 0 \times 066$
Temp(28°C)	$Id = 0 \times 0BB$	$Id = 0 \times 055$
Temp(28°C)	$Id = 0 \times 0CC$	$Id = 0 \times 044$
Temp(28°C)	$Id = 0 \times 0DD$	$Id = 0 \times 033$
Temp(28°C)	$Id = 0 \times 0EE$	$Id = 0 \times 022$
Temp(28°C)	$Id = 0 \times 0FF$	$Id = 0 \times 011$

When the system was turned on for the first time, the LCD first displays the welcome message that has been written in the program for receiver node. Then, the LCD will display the magnitude of temperature or the magnitude of relative humidity measured by the DHT sensor, depending on the message id. The magnitude displayed for relative humidity were 82% and for the temperature were 28℃.

When at another times the magnitude of the relative humidity and the temperature were monitored with this system and then the results were compared with those recorded from computer screen, the following results were obtained

Tabel 2. Magnitude of Temperature & Humidity Recorded Compared with Computer Recorded

Parameter	This System Measurements	Results Recorded on Computer Screen
Temperature	25℃	24℃
Humidity	80%	74%

It is shown in Table 2 above, for temperature measurement, this CAN system's measurement records 25℃, while magnitude of 24℃ was recorded on the computer screen. And for relative humidity measurement, this system's measurement records 80% and the computer screen records 74%. There were therefore some errors of approximately 4% and 7.5% for temperature measurement and relative humidity measurement, respectively

Still from Table 1, it can be seen that the LCD always diplays messages with a lower identification number. For example, when the humidity message has an $Id = 0 \times 001$ ($Id = 000\ 0000\ 0001$) and the temperature message has an $Id = 0 \times 0FF$ ($Id = 000\ 1111\ 1111$), the LCD displays the humidity data with magnitude 82%. On the other hand, when the humidity message has an $Id = 0 \times 0FF$ ($Id = 000\ 1111\ 1111$) and temperature message has an $Id = 0 \times 011$ ($Id = 000\ 0001\ 0001$), the LCD displays the temperature data with magnitude 28℃. Thus, the lower the binary message identifier number, the higher its priority. Even when the two message identification numbers are quite close together (for example, in the Table 1 humidity message $Id = 0 \times 039$ ($Id = 000\ 0011\ 1001$) and temperature message $Id = 0 \times 036$ ($Id = 000\ 0011\ 0110$)), the temperature message with its lower binary message identifier number still win the arbitration. Thus, messages which has a last identifier bit as a zero (dominant) is always displayed. See the difference between the four LSBs 1001 for humidity message and 0110 for temperature message.

However, when a delay was added at the end of the transmitter program (either humidity transmitter program or temperature transmitter program), something different was observed at the LCD. That is, if we add a delay statement after the statement for sending the message, (i.e. put a delay statement `delay(1000);` after the line `mcp2515.sendMessage(&canMsg);`), the message which lost arbitration will sometimes be displayed. The LCD will for a while displaying the lost arbitration message, and then back again displaying the winning message.

Figure 14 below shows one example of LCD display when it is displaying a temperature data.

Figure 14. LCD Displaying Temperature Data Measured from the Vicinity of DHT 11 Sensor.

5.3. Discussions

From the results, the hardware and programming results, it was shown that the overall system works according to the initial specification and plans. The LCD was able to display continuously relative humidity value (82%) and temperature value (28℃) according to the message priority. A message with lower identification number is the highest priority message on a network. When the first time the system was turned on, after displaying the welcome message, LCD always display the message with lower identification number.

Table 3. Comparison with Other Research Results

Parameter	Other Research		This Research
	Xu Yan et al [3]	*Q.Zhu et* [4]	
Sensor	SHT75 Temp & Humidity	Temp & Humidity	DHT11 Temp & Humidity sensor in several locations
Node Controller	C8051F060 MCU	AT91SAM7X256 MCU	ATMega 328 MCU
Display	Computer	-	LCD
Bus	CAN Bus	CAN Bus	CAN Bus
Results	Data shown on Computer	-	Data shown on LCD

In essence, the results show that after the DHT11 digital temperature and humidity sensors can measure the temperature and humidity, and those data were succesfully sent to the receiver via the CAN controller, the CAN transceiver, and the CAN bus. The system has also been proved stable and reliable.

For comparison purposes, various results from some other researches are also included, as shown in Table 3, Table 4, and Table 5.

It can be shown in those three tables, that there are several parameters that distinguish this research from other studies. Such as the type of sensors used, the node controllers, and type of display.

Table 4. Comparison with Other Research Results

Parameter	Other Research		This Research
	Li Hong Zhang et al [5]	*Hyun Lee* [16]	
Sensor	Temperature sensor	Various sensors on UAV	DHT11 Temp & Humidity sensor in several locations
Node Controller	Zigbee network STM32 WBA	-	ATMega 328 Microcontroller
Display	Computer	-	LCD I2C
Bus	CAN Bus	CAN Bus	CAN Bus
Results	Data shown on Computer	-	Data shown on LCD

Table 5. Comparison with Other Research Results

Parameter	Other Research		This Research
	T.P. Presi [17]	Guoqing Lu et al [18]	
Sensor	Temp, CO level, battery voltage	Temp & Humidity sensors in warehouse	DHT11 Temp & Humidity sensor in several locations
Node Controller	PIC MCU	Microcontroller	ATMega 328 Microcontroller
Display	-	Computer monitor	LCD I2C
Bus	CAN Bus	CAN Bus	CAN Bus
Results	-	Data on Computer monitor	Data shown on LCD

In Table 4 for example, Hyun Lee's work is shown [16] which proposes UAV (Unmanned Aerial Vehicle) engine control monitoring system using a dynamic ID application and a scheduling method of CAN network sensors which collect the temperatures, pressure, vibration, and fuel level of UAV engine through the network.

While in Table 5 are the results of T.P. Presi's work [17]. It is a project that is aimed at the implementation of CAN protocol using PIC for vehicle monitoring system. The main feature of the system includes monitoring of various vehicle parameters such as temperature, presence of CO level in the exhaust, and battery voltage. And Guoqing Lu et al's results [18] which presents a new method of the monitoring and controlling system for the great warehouse. Where the parameters such as temperature and humidity can be get, modified and displayed.

Thus, there some other researches which are similar with this work, in that they are also using this CAN network. The differences are in the measured objects (applications), the type of sensors used, the processors and the CAN electronic chips used, and also the device for displaying the results

In the future, this monitoring system in this research could be modified such that the number of transmission nodes becomes more than two nodes. In other word, a number of measurement points were added along the bus. The variable to be measured could be made uniform. For example, all consists of temperature measurements or all of humidity measurements. Thus, we could have single variable (single parameter) measurement with a lot of measuring points.

Another option could be for example, some of the measurements can be made as temperature measurements, some other are humidity measurements, and the remaining are of pressure measurements and flow measurements. This time, we have several parameter measurements with a lot of measuring points. In both cases, there are now more than two messages with their respective message identification numbers

6. Conclusions

From the experiment, it can be concluded that the overall system functions according to specifications. The temperature node can transmit its temperature data along the bus. The humidity node can also transmit its humidity data along the bus. The receiver node can then display those data with an LCD. In essence, the CAN bus is a fieldbus network that can connect many field devices together in a plant network. This research has successfully utilized this CAN bus in designing the temperature and humidity monitoring systems consisting of several nodes. Only by using a pair of cable, many field devices that typically used in process control applications can be connected together and can be made to communicate with each other. This is advantageous in that it is simple and low cost since all nodes communicate via a single CAN system instead of via direct complex analogue signal lines - reducing errors, weight, wiring and costs. Another advantage of this CAN bus system is that the system designer can determine the allocation of message priority. Some designers maybe agree on the significance of certain messages, so prioritize them. And finally, the most important thing is that this system is stable and reliable.

References

[1]. Curtis D. Johnson. Process Control Instrumentation Technology. Sixth Edition. Columbus, Ohio: Prentice Hall, 2000, pp. 165-166.

[2]. J. da Silva Sa, J. J. da Silva, M. G. Wanzeller and J. S. da Rocha Neto, "Monitoring of temperature using smart sensors based on CAN architecture," 15th International Conference on Electronics, Communications and Computers (CONIELECOMP'05), Puebla, Mexico, 2005, pp. 218-222, doi: 10.1109/CONIEL.2005.51.

[3]. Xu Yan, Guo Tao, Zhu Jie and Chen Wei, "Based on single-chip microcomputer temperature and humidity data acquisition system design," Proceedings of 2011 International Conference on Electronics and Optoelectronics, Dalian, 2011, pp. V2-310-V2-313, doi: 10.1109/ICEOE.2011.6013243.

[4]. Q. Zhu, D. Zhu and X. Su, "Distributed remote temperature monitoring and acquisition system based on CAN bus," 2010 Prognostics and System Health Management Conference, Macao, China, 2010, pp. 1-4, doi: 10.1109/PHM.2010.5413439.

[5]. L. Zhang, L. Sun and W. Lu, "A Temperature Monitoring System of Power Cable Joints Based on the Combining of CAN Wired Transmission and ZigBee Wireless Network," 2010 2nd International Conference on Information Engineering and Computer Science, Wuhan, China, 2010, pp. 1-4, doi: 10.1109/ICIECS.2010.5677661.

[6]. Franklyn W Kirk, Thomas A. Weedon, Philip Kirk. Instrumentation. Fifth Edition. USA: American Technical Publishers Inc, 2010, pp. 346 - 347

[7]. Steve Corrigan. *Introduction to the Controller Area Network (CAN)*. Texas Instrument. Application Report. Report number: SLOA101B. 2002.

[8]. Wolfhard Lawrenz. CAN System Engineering – From Theory to Practical Applications. Second Edition Wolfenbuttel, Germany: Springer, 2013, pp. 8 – 9.

[9]. Marco Di Natale, Haibo Zeng, Paolo Giusto, Arkadeb Ghosal. Understanding and Using the Controller Area Network Communication Protocol - Theory and Practice. First Edition. Palo Alto, CA: Springer, 2012, pp. 5 – 9.

[10]. Arduino CC. *Arduino® R3*. Arduino CC, Product Reference Manual. Number: SKUA000066. 2023

[11]. Arduino CC. *Arduino® Nano*. Arduino CC. Product Reference Manual. Number: SKUA000005, 2023

[12]. Microchip. MCP2515 Stand-Alone CAN Controller with SPI Interface. Microchip Technology Inc. Product Manual. Number: DS20001801J. 2018

[13]. Phillips Semiconductors. TJA1050 High Speed CAN Transceiver. Phillips Semiconductors. Product Specification. 2003

[14]. Mouser Electronics. DHT11 Humidity & Temperature Sensor. Mouser Electronics. Data Sheet.

[15]. Texas Instruments. *AN903-A Comparison of Differential Termination Techniques*. Texas Instruments. Application Report. Report number: SNLA034B. 2013

[16]. H. Lee, "UAV Engine Control Monitoring System based on CAN Network," 2020 20th International Conference on Control, Automation and Systems (ICCAS), Busan, Korea (South), 2020, pp. 820-823, doi: 10.23919/ICCAS50221.2020.9268244.

[17]. T. P. Presi, "Design and development Of PIC microcontroller based vehicle monitoring system using Controller Area Network (CAN) protocol," 2013 International Conference on Information Communication and Embedded Systems (ICICES), Chennai, India, 2013, pp. 1070-1076, doi: 10.1109/ICICES.2013.6508232.

[18]. G. Lu, G. Gao, L. Hu, R. Hao and D. Liu, "Designed of Remotely Monitoring System of Great Warehouse Based on CAN Bus," 2008 International Conference on MultiMedia and Information Technology, Three Gorges, China, 2008, pp. 420-422, doi: 10.1109/MMIT.2008.116.