"Tema: 4 (Teknik dan Energi baru dan terbarukan)"

SISTEM MONITORING DATA SUHU DAN KELEMBAPAN RELATIF BERBASIS CAN BUS

Arief Wisnu Wardhana[1], Agung Mubyarto[2] dan Acep Taryana[3]

[1]Fakultas Teknik, Universitas Jenderal Soedirman, Indonesia
[2]Fakultas Teknik, Universitas Jenderal Soedirman, Indonesia
[3]Fakultas Teknik, Universitas Jenderal Soedirman, Indonesia

ABSTRAK

Pengukuran suhu dan kelembapan merupakan aktivitas yang sering sekali perlu untuk diadakan. Ada banyak bidang yang memanfaatkan kegiatan pengukuran suhu dan kelembapan ini. Misalnya pada berbagai industri, area pertanian dan peternakan atau pada penelitian. Pengukuran bahkan sering diperlukan secara otomatis, kontinyu, dan akurat. Artikel ini memaparkan riset tentang sebuah sistem pemonitor suhu dan kelembapan relatif. Alat ini mampu untuk mencatat data suhu dan kelembapan relatif secara otomatis dan terus menerus selama 24 jam per hari. Sistem ini berbasis pada standar bus CAN (Controller Area Network), merupakan sebuah standar yang didesain agar memungkinkan banyak node kendali untuk berkomunikasi satu sama lain tanpa komputer host. Terdiri dari sebuah jalur bus CAN, dua node transmisi, serta satu node penerima. 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. Satu node transmisi mengirimkan data suhu dan satunya lagi mengirimkan data kelembapan yang sudah diukur oleh DHT11. Data kemudian diproses oleh node transmitter dan dikirimkan melalui CAN bus. Untuk menampilkan data dilakukan oleh node receiver. Beberapa pesan suhu dan pesan kelembapan dengan nomor identifikasi pesan yang berbeda beda dicoba untuk ditransmisikan. Hasilnya menunjukkan bahwa LCD selalu menampilkan pesan yang mempunyai nomor identifikasi lebih rendah. Dengan sedikit penambahan pada program untuk node transmitter, bisa dibuat data suhu terukur dan kelembapan terukur ditampilkan secara bergantian dan kontinyu pada LCD. Secara keseluruhan, dapat disimpulkan bahwa sistem ini sudah berfungsi dengan baik sesuai dengan spesifikasi awal. CAN bus yang terklasifikasi sebagai sebuah jaringan industri adalah merupakan jaringan bus perangkat yang sangat berguna. Jaringan bus perangkat ini bisa mentransfer beberapa byte informasi (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. Sistem ini juga terbukti stabil dan dapat diandalkan.

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

ABSTRACT

Measurement of temperature and humidity is an activity that is very often need to be taken. There are many fields that take advantage of this temperature and humidity measurement activity. For example, in various fields of industries, agriculture or in research. Measurements are often required automatically, accurately, and on a continuous basis. This paper presents research about a temperature and relative humidity monitoring device. The system is able to record temperature and relative humidity data automatically and continuously for 24 hours daily. The network is based on the Controller Area Network (CAN) bus standard, which is a standard designed to allow multiple controller nodes to communicate with each other without a host computer. The system consists of a CAN bus line, two transmitter nodes carried out by one Arduino UNO board and one Arduino Nano board, one receiver node carried out 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 displaying the temperature and humidity data. The DHT11 sensors measure temperature and humidity in the vicinity. There are two transmitter nodes, one transmitting temperature data and the other transmitting humidity data measured by DHT11. The data is then be processed by the transmitter nodes and sent via CAN bus. Display of the data is carried out by a node act as a receiver. Several different message identification numbers both for temperature messages and humidity messages were transmitted. The results show that the LCD always display the messages that have lower message identification number. With a few additions to the programs for the transmitter nodes, measured temperature and humidity data can be displayed alternately and continuously on the LCD. In the overall, it can be concluded that the system function according to initial specifications. CAN bus which is classified as an industrial network is a very versatile device bus network. This device bus network can transfer a few bytes (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 id, fieldbus network

PENDAHULUAN

Ilmu pengetahuan dan teknologi dewasa ini telah berkembang sangat pesat terutama dalam hal-hal yang dapat membantu kehidupan dan pekerjaan manusia sehingga menjadi lebih mudah dan efisien. Manusia telah berkembang mulai dari menggunakan metode primitif hingga mencapai kondisi modern seperti sekarang ini dalam cara mereka menjalani kehidupan sehari-hari. Beberapa contoh yang menunjukkan aktivitas manusia dalam kehidupannya sehari-hari, adalah dalam menggunakan berbagai alat untuk membantunya, mengolah bahan makanan untuk kebutuhan makanannya, atau bahkan sekedar untuk merasakan dan mengamati cuaca di sekitarnya.

Misalnya, seseorang mungkin hanya sekedar merasakan dan mengamati suhu atau kelembapan di sekitarnya tanpa tujuan lebih lanjut. Kegiatan ini juga dapat dilakukan lebih jauh lagi yaitu dengan sengaja memantau suhu dan kelembaban secara terus menerus untuk tujuan tertentu. Hal ini tentunya membutuhkan alat perekam yang akurat.

Jika kita ingin mengukur suhu atau kelembaban, tentunya harus ada semacam satuan baku yang dapat digunakan untuk mengklasifikasikan pengukuran ini. Dahulu, satuan yang digunakan mungkin adalah

sekedar 'panas' dan 'dingin' [1]. Dan hanya istilah 'lembab' dan 'kering' yang digunakan untuk mengklasifikasikan pengukuran kelembapan. Ini mungkin cukup untuk waktu itu, tetapi ini tentu tidak memadai untuk penggunaan masa sekarang. Selain itu, karena sifat pengukurannya, terkadang diperlukan pemantauan terus menerus. Misalnya, kita sering perlu mencatat (atau memantau) suhu atau kelembapan yang berubah setiap menit atau bahkan detik. Jika dilakukan secara manual tentu akan menjadi tidak efisien karena memakan banyak waktu dan mempersulit pengumpulan data. Oleh karena itu, diperlukan sistem pemantauan data yang mudah digunakan.

Berbagai jenis sistem pemantauan data suhu dan kelembaban dapat dirancang. Mungkin dalam bentuk pemantauan data suhu dan kelembaban yang berdiri sendiri yang terdiri dari satu perangkat. Di sisi lain, bisa juga berupa kumpulan dari beberapa perangkat yang terhubung dalam jaringan. Untuk yang jenis jaringan, suatu metode (disebut standar atau protokol) tentu diperlukan untuk mengatur transmisi data antar perangkat. Beberapa protokol telah dibakukan, bahkan sudah masuk dalam jaringan industri seperti standar MODBUS, standar Profibus, standar CAN bus.

Banyak penelitian tentang pemantauan suhu dan kelembaban telah dilakukan. Beberapa di antaranya terdiri dari beberapa perangkat sensor yang terhubung dalam jaringan. Yang dirancang berdasarkan bus CAN standar dapat disebutkan beberapa diantaranya adalah sebagai berikut.

Pertama, ada penelitian yang dilakukan oleh J. da Silva Sa et al mempresentasikan implementasi sensor pintar untuk memantau suhu dan untuk berkomunikasi di antara mereka menggunakan protokol CAN [2]. Selanjutnya, yang dilakukan oleh Xu Yan et al memperkenalkan desain sistem akuisisi data yang terdiri dari sensor suhu dan kelembaban, sistem SCM, komputer, bus CAN [3]. Ada juga penelitian lain yang dilakukan oleh Q. Zhu et al mengusulkan platform sistem tertanam pemantauan suhu jarak jauh. Mikroprosesor tertanam AT91SAM7X256 digunakan sebagai CPU sistem. Sistem ini mewujudkan pemantauan dan penyimpanan pengumpulan data jarak jauh secara real-time melalui konversi data protokol dari bus CAN dan bus RS232 dari node akuisisi suhu terdistribusi [4]. Dan penelitian yang bertujuan untuk memantau karakteristik lingkungan, dirancang sistem pemantauan suhu sambungan kabel listrik on-line berdasarkan kombinasi transmisi kabel CAN dan jaringan nirkabel ZigBee, yang dilakukan oleh Lihong Zhang et al [5].

Jaringan alat pemantau suhu dan kelembaban yang dirancang pada penelitian ini juga berdasarkan standar CAN bus yaitu Standar CAN2.0b. Sistem ini terdiri dari tiga node CAN lengkap dengan masing-masing pengontrol CAN dan transceiver CAN nya. Dua node akan melakukan transmisi data suhu dan kelembaban relatif melalui bus, dan satu node lagi akan melakukan tugas menerima dan menampilan data. CAN standar dengan pengidentifikasi 11-bit akan digunakan sebagai format bingkai data CAN.

Alat ini mampu memantau suhu dan kelembapan di sekitar lokasi sensor. Sensor suhu dan kelembaban dapat ditempatkan di beberapa lokasi berbeda. Pada percobaan ini dipasang dua buah sensor kelembaban dan suhu pada dua lokasi yang berbeda. Namun, karena ini adalah sistem terbuka, maka dimungkinkan untuk memiliki hingga 8 node perangkat pada satu bus CAN karena pada CAN standar dengan pengidentifikasi 11-bit ini, hingga 8 byte data dapat ditransmisikan. Dengan kata lain, dimungkinkan untuk diadakan penambahan lagi beberapa titik pengukuran. Contoh benda yang suhu dan kelembapan relatifnya dapat diukur dengan alat ini terutama adalah ruangan yang peka terhadap perubahan suhu dan kelembapan relatif.

Untuk kabel maksimum dan jumlah node maksimum yang dimungkinkan untuk bus CAN ini, panjang bus bisa sampai 40 m dan maksimum 30 node.

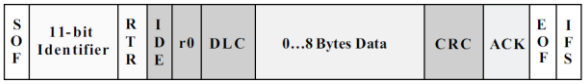
BUS CONTROLLER AREA NETWORK (CAN)

Bus CAN adalah jaringan bus perangkat berdasarkan teknologi chip elektronik CAN yang banyak digunakan. Awalnya digunakan pada kendaraan untuk mengontrol berbagai komponen internal nya seperti rem dan sistem lainnya [6]. Bus CAN dikembangkan oleh BOSCH sebagai sebuah sistem pengiriman pesan multi-master dengan tingkat pensinyalan maksimum 1 megabit per detik (bps) [6]. Tidak seperti jaringan tradisional seperti USB atau Ethernet, CAN tidak mengirimkan data dalam bentuk blok besar *point-to-point* dari node A ke node B di bawah pengawasan master bus pusat. Dalam jaringan CAN, pesan pesan singkat (hingga delapan byte) seperti suhu, kelembapan, atau RPM disiarkan ke seluruh jaringan, yang menyediakan konsistensi data di setiap node sistem.

CAN Standar

Protokol komunikasi CAN adalah sebuah *carrier-sense, multiple-access protocol with collision detection and arbitration on message priority (CSMA / CD+AMP)*. CSMA berarti bahwa setiap node pada bus harus menunggu suatu periode tidak aktif yang telah ditentukan, sebelum mencoba mengirim pesan. CD+AMP berarti bahwa tabrakan antar pesan diselesaikan melalui arbitrase bit-wise, berdasarkan prioritas yang telah diprogram sebelumnya pada setiap pesan, yaitu pada bagian pengidentifikasi pesan. Pengidentifikasi dengan prioritas lebih tinggi selalu memenangkan akses bus. Artinya, logika-tinggi terakhir (dalam CAN, logika-tinggi diasosiasikan dengan nol) dalam pengidentifikasi terus ditransmisikan karena merupakan prioritas tertinggi.

Standar CAN ISO-11898:2003, dengan standar pengidentifikasi 11-bit, menyediakan kecepatan pensinyalan dari 125 kbps hingga 1 Mbps. Bidang bit Standar CAN ditunjukkan pada Gambar 1 di bawah ini [7].



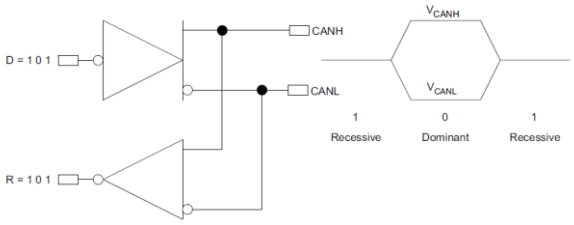
Gambar 1. CAN standar dengan pengidentifikasi 11-bit

CAN standar dengan pengidentifikasi 11-bit menentukan prioritas pesan. Semakin rendah nilai binernya, semakin tinggi prioritasnya. Data – hingga 8 x 8 byte =64 bit data aplikasi dapat ditransmisikan.

Pesan CAN

Akses bus CAN diatur dengan metode *nondestructive bitwise arbitration*. Nondestruktif adalah berarti bahwa frame yang menjadi pemenang arbitrase, yaitu pesan dengan prioritas lebih tinggi, tidak terganggu dan tidak perlu di-restart. Mekanisme ini membutuhkan driver fisik yang relevan untuk diimplementasikan dengan cara tertentu: Dua level logika pada bus CAN harus dominan dan resesif, artinya sutu node, yang mengirimkan level dominan, akan menimpa semua node lain yang mengirim level resesif. Dalam protokol CAN, sebuah logika satu dikirim resesif dan sebuah logika nol dikirim dominan.

Karakteristik CAN fundamental yang ditunjukkan pada Gambar 2 adalah keadaan logika yang berlawanan antara bus, dan input driver dan output penerima. Biasanya, logika-tinggi diasosiasikan dengan satu, dan logika-rendah diasosiasikan dengan nol - tetapi tidak demikian pada bus CAN [7].

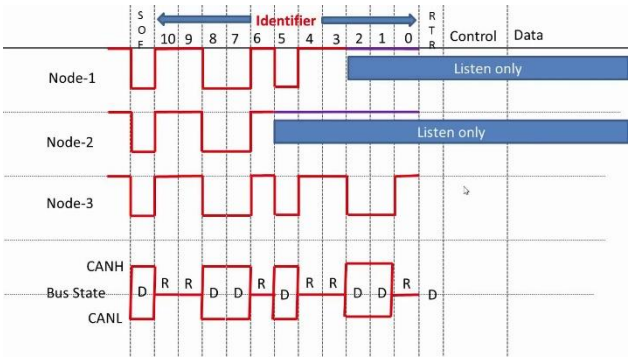


Gambar 2. Logika terbalik dari sebuah bus CAN

Akses bus dilakukan secara *event-driven* dan dilakukan secara acak. Jika terdapat dua node yang mencoba untuk menempati bus secara bersamaan, akses diimplementasikan dengan cara *non-destructive, bit-wise arbitration. Non-destructive* berarti bahwa node pemenang arbitrasi tetap terus melanjutkan pesan nya, dan pesan tersebut tidak akan dihancurkan atau dirusak oleh node lain. Semakin rendah nilai biner dari pesan *id*, semakin tinggi prioritas pesan tersebut. Jadi, pesan dengan pengidentifikasi yang seluruhnya terdiri dari nol adalah pesan prioritas tertinggi di jaringan karena memegang bus dominan paling lama. Oleh karena itu, jika dua node mulai mentransmisikan secara bersamaan, node yang mengirimkan bit pengidentifikasi terakhir sebagai nol (dominan) sementara node lain mengirim bit satu (resesif) akan mempertahankan kendali bus CAN dan melanjutkan untuk menyelesaikan pesannya. Bit dominan selalu menimpa bit resesif pada bus CAN [7].

Contoh di bawah ini mengilustrasikan arbitrasi antara tiga node CAN yang bersaing mengendalikan bus CAN (lihat Gambar 3). Gambar 3 tersebut menampilkan proses arbitrase CAN yang ditangani secara otomatis oleh pengontrol CAN.

Terlihat tiga node CAN, node-1, node-2, dan node-3 memulai transmisi pada saat yang bersamaan. Pesan *id* 11-bit untuk node-1 adalah 110 0101 1101 (Id = 0×65D), pesan *id* untuk node-2 adalah 110 0111 0110 (Id = 0×676), dan pesan *id* untuk node-3 adalah 110 0101 1001 (Id = 0×659).



Gambar 3. Arbitrasi pada CAN

Sesuai dengan metode akses *Carrier Sense Multiple Access with Collision Detection and Arbitration on Message Priority (CSMA/CD + AMP)*, ketiga node tersebut harus menunggu sampai bus CAN bebas *(Carrier Sense)*. Setelah bus CAN ini terdeteksi bebas, ketiga node tersebut secara bersama sama mengirimkan bit *Start Of Frame (SOF)* mereka *(Multiple Access)*. Bit *SOF* tersebut bernilai *dominant*. Perhatikan bahwa sebuah node transmisi akan secara terus menerus memonitor setiap bit yang dia transmisikan.

Sepanjang frame, setiap node CAN akan, melalui *transceiver* nya, membaca kembali nilai logika yang terjadi pada bus CAN dan akan membandingkannya dengan nilai logika yang telah ia transmisikan tadi *(Collision Detection)*. Ada sejumlah 3 bit *Start Of Frame* bernilai *dominant* yang ditumpangkan pada bus, dan ternyata sebuah level bus *dominant* juga terbaca kembali oleh ketiga node tersebut. Selanjutnya, Most Significant Bit (MSB) dari pesan *id* dikirim. Pada contoh di atas, nilai MSB adalah *resesif* pada ketiga pesan *id*. Di sini ternyata, level *resesif* juga muncul di bus dan sekali lagi ini dikenali oleh ketiga node. Oleh karena itu, tidak ada node yang melihat node lain sebagai *competitor* nya (hal seperti ini terjadi sampai identifier bit 6) hingga suatu saat akan sampai pada perbedaan pertama pada identifier [8].

Pada contoh di atas, perbedaan pertama antara dua frame adalah pada identifier bit 5. Node-1 dan node-3 mengirimkan level *dominant*, sementara node-2 mengirimkan level resesif. Menurut spesifikasi protokol, level *dominant* muncul di bus. Node-1 dan node-3 membaca kembali level yang telah dikirimkannya; karenanya, node-1 dan node-3 tidak melihat tabrakan. Node-2 juga membaca kembali level *dominant* dan membandingkannya dengan level resesif yang dikirimkannya; oleh karena itu, ia melihat *bit-error*. Pada titik ini, node-2 menyadari bahwa ia telah kehilangan arbitrase *(Arbitration on Message Priority)* dan node-2 segera menghentikan pengiriman framenya sendiri [8].

Perbedaan antara dua frame selanjutnya terjadi pada identifier bit 2. Node-3 mengirimkan level *dominant*, sementara node-1 mengirimkan level *resesif*. Menurut spesifikasi protokol, level *dominant* muncul di bus. Node-3 membaca kembali level yang telah dikirimnya; karenanya, ia tidak melihat tabrakan. Node-1 juga membaca kembali level *dominant* pada bus dan membandingkannya dengan level *resesif* yang dikirimkannya; oleh karena itu, ia melihat *bit-error*. Pada titik ini, node-1 menyadari bahwa ia telah kehilangan arbitrasi dan oleh karena nya, node-1 segera menghentikan pengiriman framenya sendiri [8].

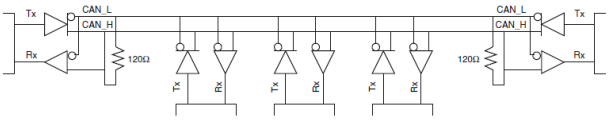
Lebih jauh lagi, sekarang node-1 dan node-2 akan menjadi penerima frame dari node-3, karena frame yang telah memenangkan arbitrasi (yaitu node-3) mungkin berisi data yang perlu diproses oleh node-1 dan node-2 [8].

Jadi, dalam contoh di atas, node-3 telah memenangkan arbitrase. Ini karena node-3 memiliki biner pesan *id* paling rendah (Id =0×659 =110 0101 1001 adalah angka *id* paling rendah). Perhatikan bahwa node-3 adalah node yang mengirimkan bit pengenal terakhir sebagai nol *(dominant)*. Lihat perbedaan antara empat LSB 1101 untuk node-1 dan empat LSB 1001 untuk node-3. Oleh karena itu, node-3 ini yang akan terus melanjutkan pesan nya.

Terminasi Bus

Sinyal listrik pada bus akan dipantulkan pada ujung jalur listrik kecuali ada tindakan pencegahan yang dilakukan. Pemantulan sinyal akan terjadi ketika sinyal ditransmisikan melalui media transmisi, seperti kabel tembaga atau serat optik. Sebagian dari daya sinyal akan dipantulkan ke asal nya, tidak dibawa sampai ke ujung yang satunya lagi. Hal ini bisa terjadi karena ketidaksempurnaan pada kabel akan menyebabkan *mismatch* dan perubahan perubahan non-linear pada karakteristik kabel. Untuk menghindari adanya *mismatch* ketika sebuah node membaca status dari bus, refleksi sinyal ini harus dihindari. Aksi pencegahan yang paling baik adalah dengan men terminasi kan lintasan bus dengan sebuah resistor terminasi pada kedua ujung lintasan, dan juga dengan menghindari adanya *long stub line* yang tidak diperlukan. Laju transmisi dan panjang bus maksimum akan tercapai dengan menggunakan sebuah struktur yang sedekat mungkin dengan jalur transmisi dan men terminasi kedua ujung nya. Metode untuk men terminasi CAN *hardware* ini bermacam macam tergantung dari *physical layer* dari *hardware* itu sendiri (kecepatan tinggi, kecepatan rendah, kawat tunggal, atau software-selectable). Untuk CAN kecepatan tinggi, kedua ujung pasangan kabel sinyal (CAN H dan CAN L) harus di terminasi. Besarnya resistor terminasi pada kabel harus sesuai dengan impedansi nominal kabel [9].

ISO 11898 mensyaratkan kabel dengan impedansi nominal 120 Ω, dan oleh karenanya resistor 120 Ω harus digunakan untuk terminasi. Jika ada beberapa perangkat ditempatkan di sepanjang kabel CAN, hanya perangkat di ujung kabel yang membutuhkan resistor terminasi [9].

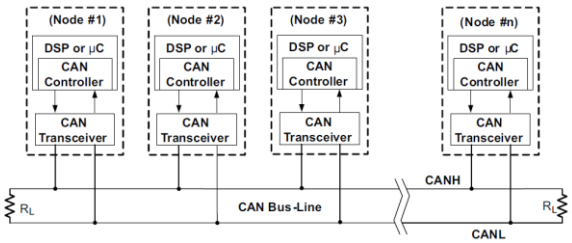


Gambar 4. Cara men terminasi sebuah jaringan berkecepatan tinggi

Gambar 4 di atas memberikan sebuah contoh cara men terminasi jaringan CAN berkecepatan tinggi. Terminasi dapat berupa tipe dasar (resistor tunggal seperti pada gambar 4 tersebut) atau tipe split (dua resistor 60 Ω dengan kapasitor perantara antara 10 nF dan 100 nF yang terhubung ke ground) [9].

Bus CAN Detail

Data link dan *physical signalling layer*, yang biasanya terlihat oleh operator sistem, sudah tersedia dalam kontroler (berbentuk kontroler embedded atau kontroler berdiri sendiri) yang mengimplementasikan protokol CAN. Koneksi ke *physical medium* kemudian diimplementasikan melalui sebuah *line transceiver* untuk membentuk sebuah node sistem seperti yang ditunjukkan pada Gambar 5 di bawah ini [7].



Gambar 5. Bus CAN secara detail

Overview Sistem

Sistem pemantauan suhu dan kelembaban relatif yang dirancang di sini akan terdiri dari antarmuka bus CAN yang terdiri dari jalur bus CAN, beberapa node CAN, beberapa pengontrol CAN, beberapa transceiver CAN, beberapa sensor, satu perangkat tampilan, dan dua resistor terminasi.

Jalur bus CAN akan berbentuk bus dua jalur. Satu jalur adalah untuk jalur CAN bus High dan yang satunya lagi adalah untuk jalur CAN bus Low.

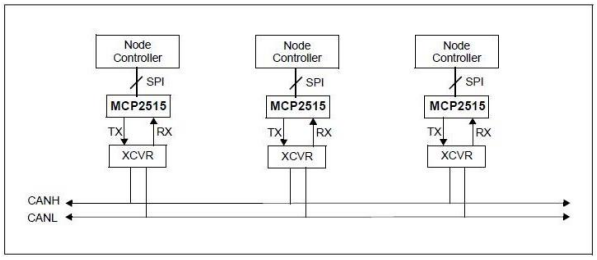
Node pertama adalah node penerima. Node ini akan dijalankan oleh Arduino UNO R3. Board Arduino UNO R3 ini adalah papan yang sempurna untuk pembelajaran elektronik dan pengkodingan. Mikrokontroler serbaguna ini dilengkapi dengan ATmega328P dan Prosesor ATMega 16U2 [10]. Node selanjutnya merupakan node transmisi untuk data temperatur. Ini juga akan dijalankan oleh papan Arduino UNO. Dan node ketiga merupakan node transmisi untuk data kelembaban. Node kelembaban ini akan dilakukan oleh Arduino® Nano. Papan ini adalah papan pengembangan cerdas yang dirancang untuk membuat prototipe yang lebih cepat dengan dimensi terkecil. Inti dari board ini adalah mikrokontroler ATmega328 yang memiliki clock frekuensi 16 MHz. Board ini menawarkan 22 pin input/output digital, 8 pin analog, dan port mini-USB [11].

Untuk pengontrol CAN, digunakan pengontrol CAN chip MCP2515. MCP2515 dari Microchip

Technology adalah pengontrol Controllen Area Network (CAN) yang berdiri sendiri yang mengimplementasikan spesifikasi CAN, Versi 2.0B. MCP2515 mampu mentransmisikan dan menerima data standar dan tambahan dan bingkai jarak jauh. MCP2515 memiliki dua *mask* penerimaan dan enam filter penerimaan yang digunakan untuk menyaring pesan yang tidak diinginkan, sehingga mengurangi overhead MCU host. MCP2515 berinteraksi dengan mikrokontroler (MCU) melalui standar industri *Serial Peripheral Interface (SPI)* [12].

Berikutnya adalah transceiver CAN. Transceiver CAN ini mentransmisikan dan menerima data fisik ke bus dan dari bus. Transceiver chip TJA1050 digunakan di sini. TJA1050 adalah antarmuka antara pengontrol CAN dan fisikal bus CAN. Perangkat ini memberikan kemampuan transmisi diferensial ke bus dan kemampuan penerimaan diferensial ke pengontrol CAN. Ini sepenuhnya kompatibel dengan standar "ISO 11898" [13].

Digambarkan dalam diagram, interface bus CAN dari sistem ini akan terlihat seperti Gambar 6 di bawah ini.



Gambar 6. Antarmuka Bus CAN dari sistem pemonitor suhu dan kelembapan

Pada Gambar 6 di atas, bagian blok yang ditunjukkan oleh *node controller* adalah papan papan Arduino dengan sensor suhu & kelembaban dan layar LCD nya. Sedangkan blok yang ditunjukkan oleh XCVR adalah transceiver CAN TJA1050.

Seperti disebutkan di atas, sistem juga akan memiliki serangkaian perangkat untuk berkomunikasi dengan dunia luar. Perangkat tersebut adalah sensor dan perangkat tampilan. Untuk sensor digunakan sensor kelembaban dan suhu DHT11. Sensor suhu & kelembaban DHT11 adalah sebuah sensor suhu & kelembaban kompleks dilengkapi dengan output sinyal digital yang dikalibrasi. Dengan menggunakan teknik akuisisi sinyal digital eksklusif dan teknologi penginderaan suhu & kelembaban, sensor ini memastikan keandalan yang tinggi dan stabilitas jangka panjang yang sangat baik. Sensor ini mencakup komponen pengukuran kelembaban tipe resistif dan komponen pengukuran suhu NTC, dan bisa terhubung ke mikrokontroler 8-bit berkinerja tinggi, menawarkan kualitas luar biasa, respons cepat, kemampuan anti-interferensi, dan hemat biaya [14].

Sementara untuk perangkat layar tampilan, digunakan LCD 16×2 dengan antarmuka I2C. Ini adalah modul layar LCD 16x2 antarmuka I2C, modul LCD 2 baris 16 karakter berkualitas tinggi dengan penyesuaian kontrol kontras terpasang, lampu latar, dan antarmuka komunikasi I2C

METODE PENELITIAN

Metode yang digunakan dalam penelitian ini meliputi pemrograman ketiga node dan perakitan berbagai bagian sistem.

Untuk pemrograman, semua papan Arduino akan diprogram dengan Arduino IDE. Arduino adalah platform elektronik sumber terbuka berdasarkan perangkat keras dan perangkat lunak yang mudah

digunakan. Papan Arduino dapat membaca input - cahaya pada sensor, jari pada tombol, atau pesan Twitter - dan mengubahnya menjadi output - mengaktifkan motor, menyalakan LED, menerbitkan sesuatu secara online. Kita dapat memberi tahu board apa yang harus dilakukan nya dengan mengirimkan satu set instruksi ke mikrokontroler di board tersebut. Untuk melakukannya, kita menggunakan bahasa pemrograman Arduino (berdasarkan perkabelan), dan perangkat lunak Arduino (IDE), berdasarkan pemrosesan. Setiap papan akan diprogram sesuai dengan fungsinya. Jadi, di sini akan ada tiga program Arduino yang berbeda.

Untuk perangkat keras nya, semua bagian sistem akan dirakit sesuai diagram pada Gambar 5 dan Gambar 6. Dengan demikian, terdapat tiga node yang sudah terprogram yang akan dihubungkan ke jalur bus CAN. Dua node di ujung jalur CAN adalah node penerima dan node kelembaban. Node suhu ditempatkan di tengah jalur, menyadap jalur. Juga akan ada dua resistor yang bertindak sebagai resistor terminasi di ujung jalur.

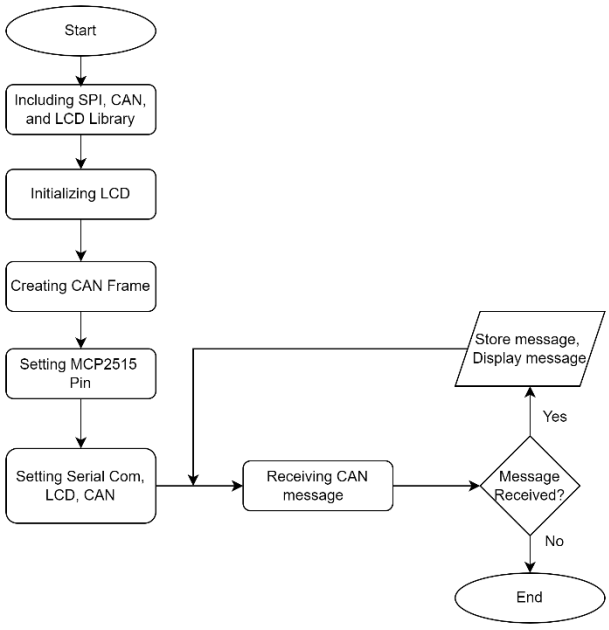
Selanjutnya, akan ada eksperimen dengan bilangan pesan *id* 11-bit. Artinya, pasangan bilangan pesan *id* (satu untuk pesan suhu dan satu lagi untuk pesan kelembapan) divariasikan beberapa kali. Untuk setiap pasangan dilakukan percobaan pengiriman data suhu dan kelembaban. Hasilnya kemudian diamati pada perangkat penampil LCD.

HASIL DAN PEMBAHASAN

Jadi, hasil penelitian ini terdiri dari dua bagian, yaitu hasil program dan hasil perangkat kerasnya. Bagian program terdiri dari tiga buah program Arduino untuk tiga buah node. Satu program untuk masing masing node. Sementara hasil hardware adalah merupakan penggabungan dari berbagai part dan hasil hasil dari tampilan LCD.

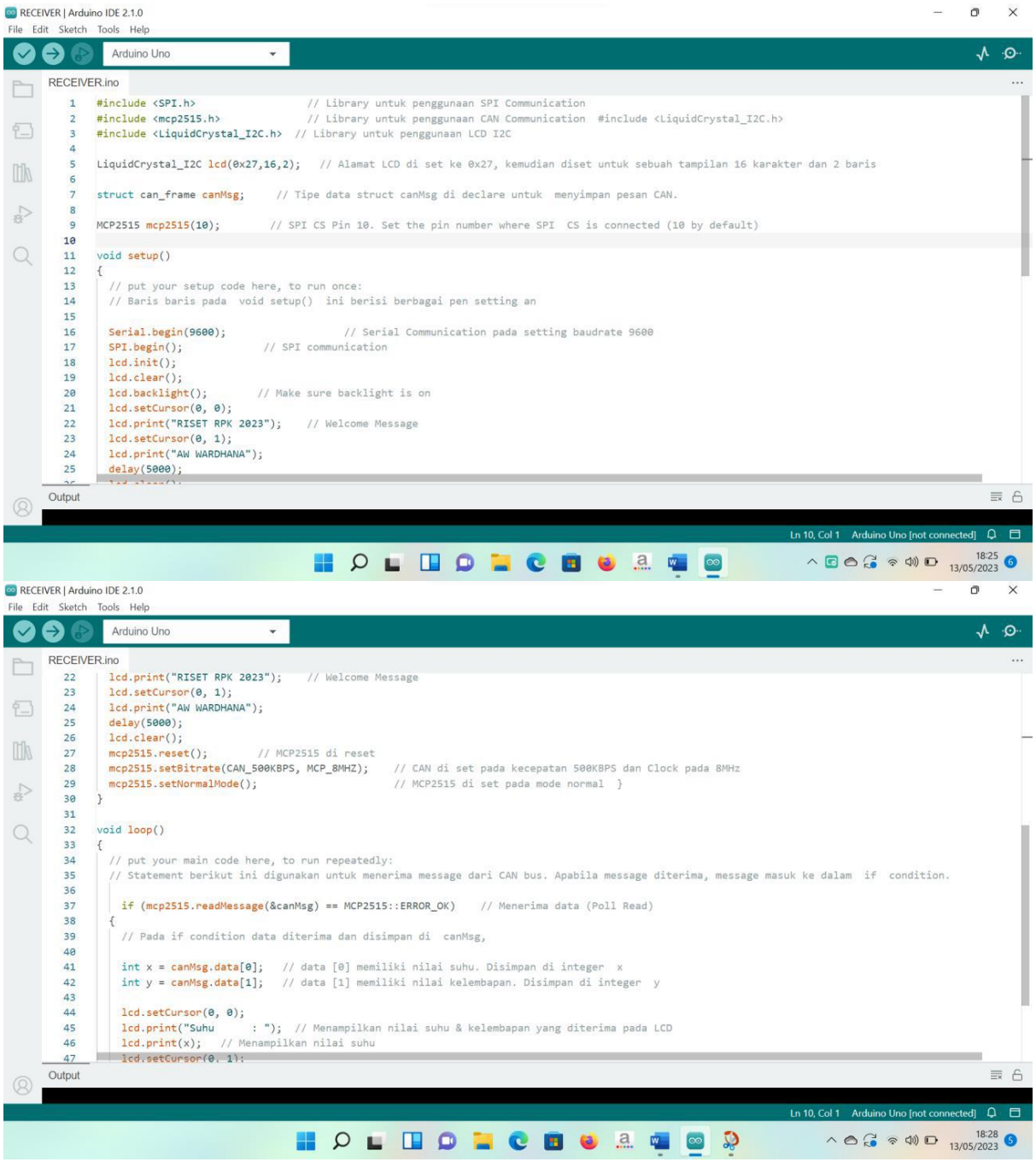
Hasil Pemrograman

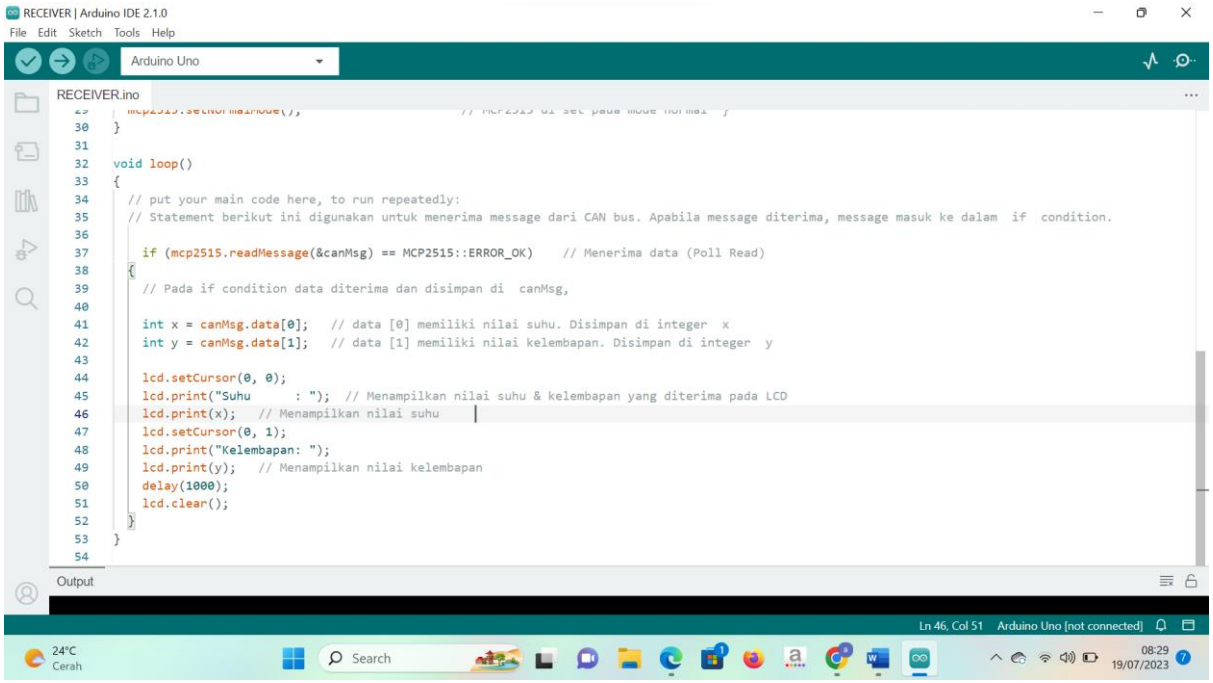
Ada tiga program Arduino yang sudah dibuat untuk ketiga node tersebut. Pertama adalah program listing untuk node receiver, yang flowchart nya ditunjukkan pada Gambar 7 di bawah ini.



Gambar 7. Flowchart untuk Program Node Receiver

Dari flow chart di atas, program listing Arduino untuk receiver dibuat seperti yang ditunjukkan pada Gambar 8. Penjelasan untuk setiap baris tersedia di sebelah kanan nya.





Gambar 8. Program Arduino untuk Program Node Receiver

Bisa ditunjukkan bahwa node receiver ini sudah diprogram sedemikian rupa sehingga node ini hanya akan bisa menerima dan menampilkan data kelembapan yang dikirim oleh node kelembapan dan juga menerima dan menampilkan data suhu yang dikirim oleh node temperatur. Node receiver ini tidak mengirimkan data.

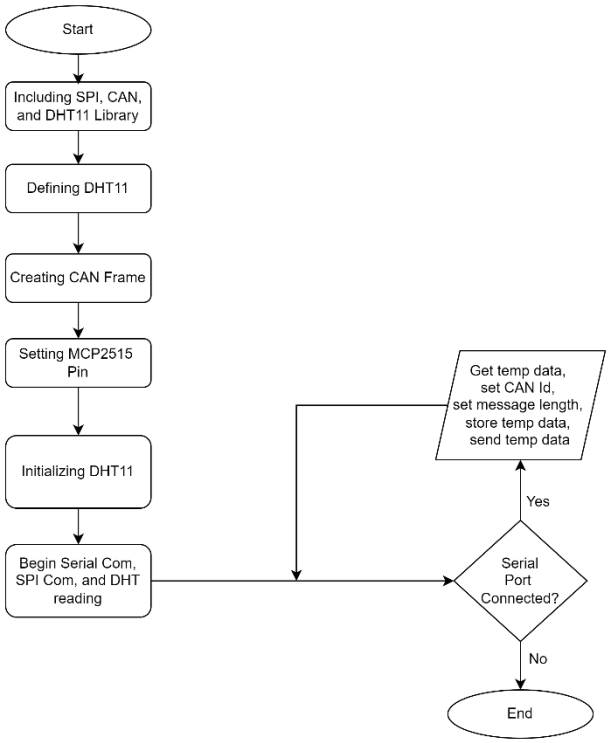
Beberapa library digunakan untuk mengimplementasikan komunikasi SPI, mengimplementasikan komunikasi CAN, dan menggunakan LCD I2C . Setelah memasukkan library, pensettingan LCD dilakukan, dan sebuah struktur untuk menyimpan pesan CAN dideklarasikan.

Di dalam void setup() adalah berbagai pensettingan untuk SPI, LCD, dan CAN. Di dalam void loop(), sebuah pernyataan untuk membaca pesan pesan CAN yang dikirimkan oleh transmiter (pesan CAN berisi data temperature dan data kelembapan) dibuat, dan integer juga dibuat untuk menyimpan data temperature dan kelembapan tersebut. Dan juga terdapat beberapa pernyataan berkaitan dengan LCD untuk menampilkan data.

Selanjutnya adalah program listing untuk node pengirim data temperature. Flowchart untuk program temperature ini ditunjukkan di Gambar 9.

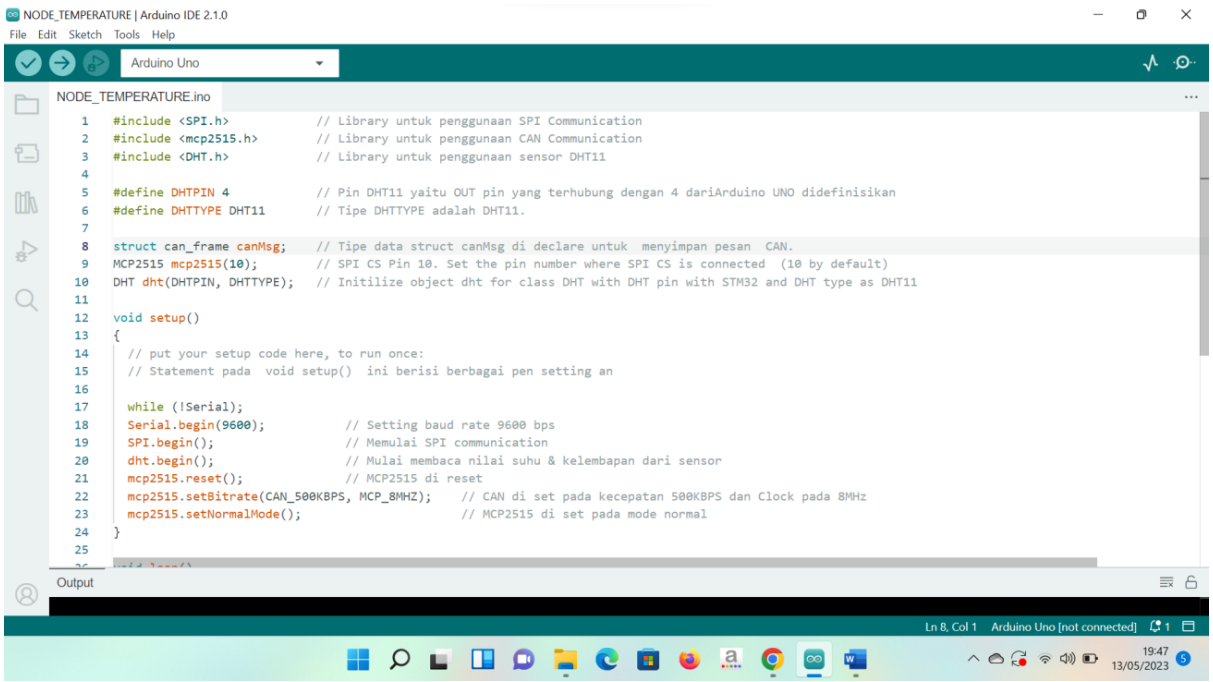
Dari flowchart tersebut, dibuat program listing Arduino untuk node temperatur seperti yang ditunjukkan pada Gambar 10. Penjelasan untuk setiap baris disediakan di sebelah kanan nya.

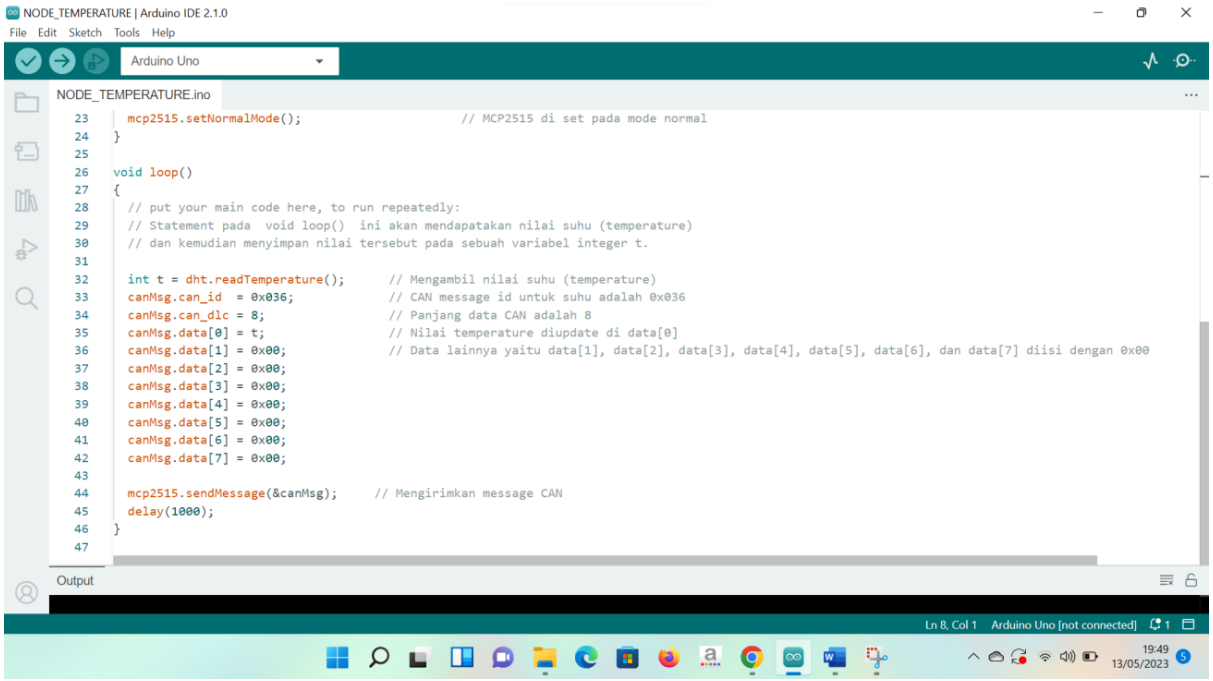
Node temperature ini sudah diprogram sedemikian rupa sehingga hanya bisa mengirimkan data temperature yang sudah dicatat oleh sensor DHT11.



Gambar 9. Flowchart untuk Program Node Temperature

Beberapa library digunakan untuk mengimplementasikan komunikasi SPI, mengimplementasikan komunikasi CAN, dan untuk penggunakan sensor DHT11.





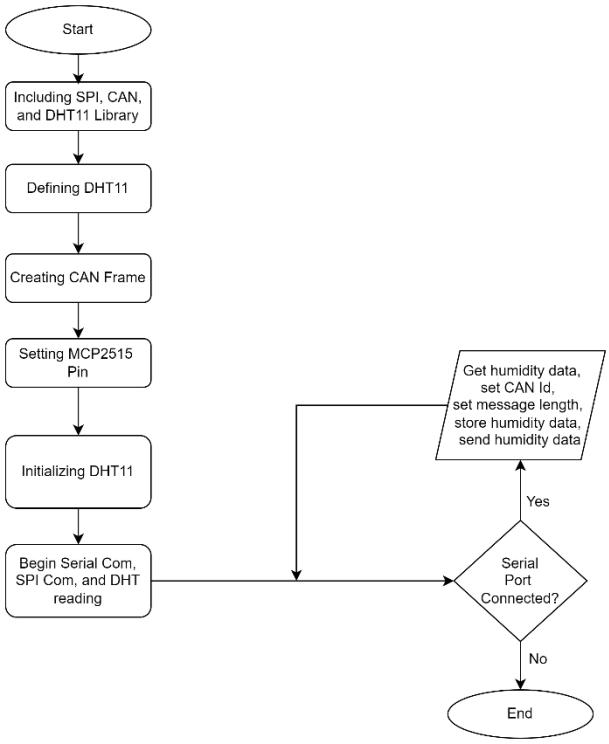
Gambar 10. Program Arduino untuk Node Temperature

Di dalam void setup() adalah berbagai pensettingan untuk SPI, DHT11, dan CAN. Di dalam void loop(), dibuat sebuah pernyataan untuk membaca temperature dan kemudian menyimpan nya pada sebuah integer. Kemudian, ditunjukkan sebuah contoh *message identifier* untuk temperature yaitu $\boldsymbol{0x036}$ $(\boldsymbol{Id = 000\ 0011\ 0110})$. Ditunjukkan juga bahwa panjang pesan CAN adalah 8 bytes, ditunjukkan sebagai canMsg.data[0],canMsg.data[1],canMsg.data[2],canMsg.data[3],canMsg.data[4],canMsg.data[5],canMsg.data[6], dan canMsg.data[7]. Masing masing panjangnya satu byte. Ini sesuai dengan struktur dari CAN standar yang mempunyai data 0 … 8 btyes. Data temperature diletakkan pada canMsg.data[0]. Sementara byte byte yang lain diisi dengan $\boldsymbol{0x00}$. Bagian void loop() ini kemudian diakhiri dengan sebuah pernyataan yang berhubungan dengan MCP2515 untuk mengirimkan pesan CAN.

Dan terakhir, adalah program listing untuk node pengirim data kelembapan. Flowchart untuk program kelembapan ini ditunjukkan di Gambar 11.

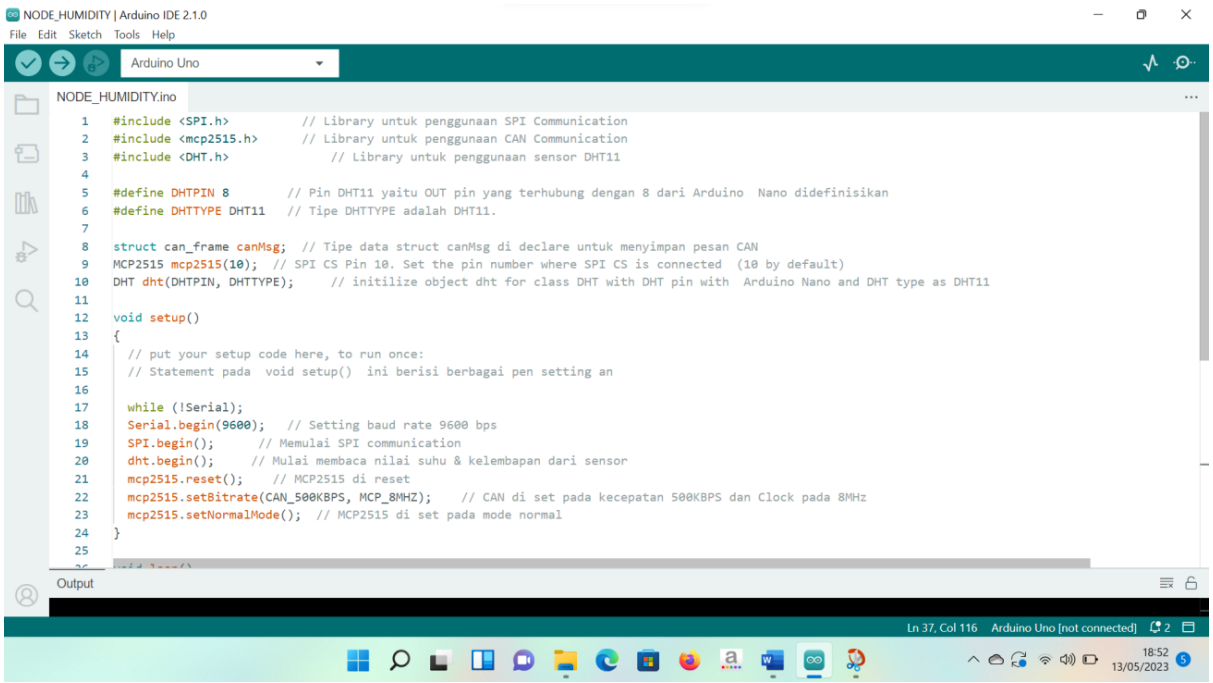
Dari flowchart tersebut, dibuat program listing Arduino untuk node kelembapan seperti yang ditunjukkan pada Gambar 12. Penjelasan untuk setiap baris disediakan di sebelah kanan nya.

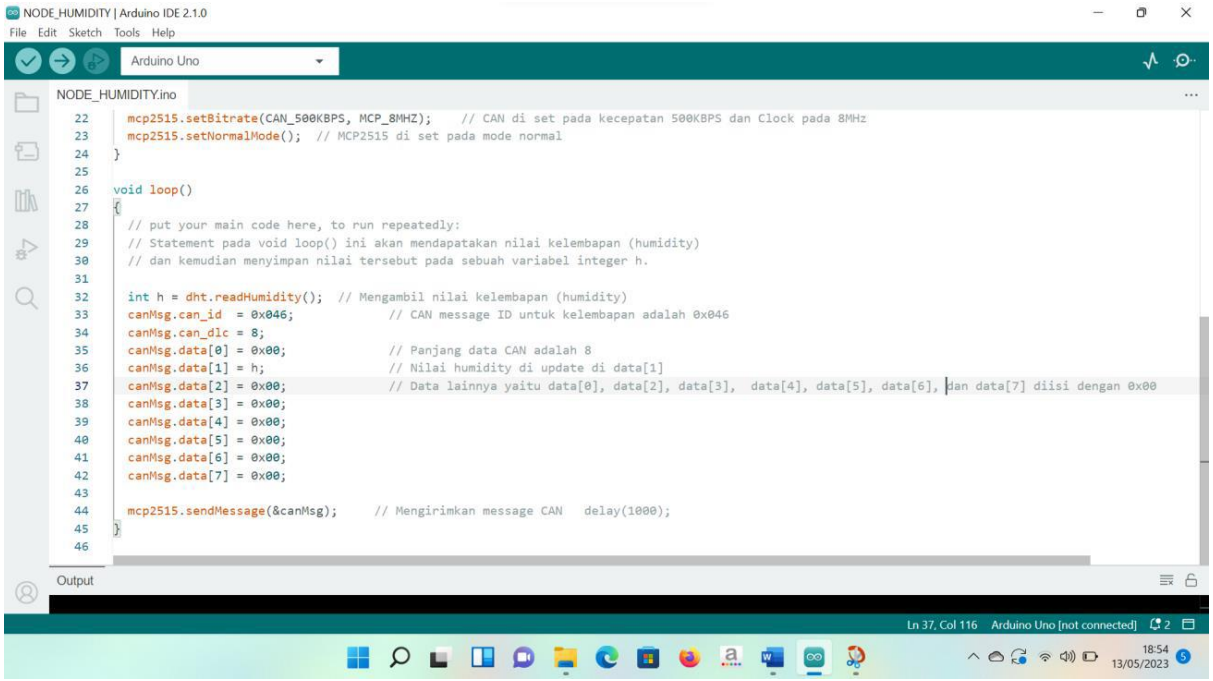
Node kelembapan ini sudah diprogram sedemikian rupa sehingga hanya bisa mengirimkan data kelembapan yang sudah dicatat oleh sensor DHT11.



Gambar 11. Flowchart untuk Program Node Kelembapan

Beberapa library digunakan untuk mengimplementasikan komunikasi SPI, mengimplementasikan komunikasi CAN, dan penggunakan sensor DHT11.



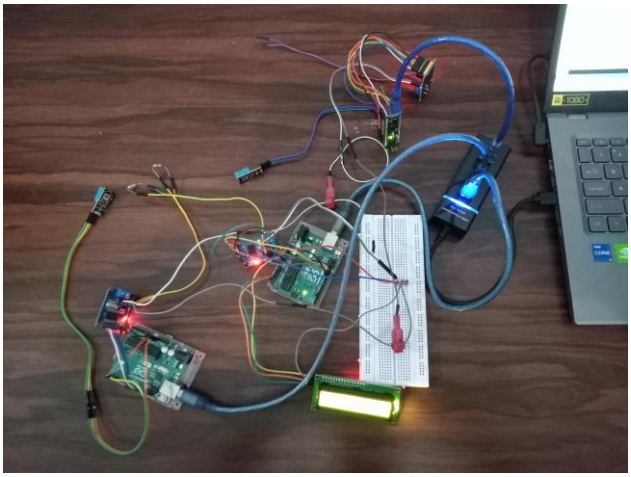


Gambar 12. Program Arduino untuk Node Kelembapan

Di dalam void setup() adalah berbagai pensettingan untuk SPI, DHT11, dan CAN. Di dalam void loop(), dibuat sebuah pernyataan untuk membaca data kelembapan dan kemudian menyimpan nya pada sebuah integer. Kemudian, ditunjukkan sebuah contoh *message identifier* untuk kelembapan yaitu $\boldsymbol{0x046}$ ($\boldsymbol{Id = 000\ 0100\ 0110}$). Ditunjukkan juga bahwa panjang pesan CAN adalah 8 bytes, ditunjukkan sebagai canMsg.data[0],canMsg.data[1],canMsg.data[2],canMsg.data[3],canMsg.data[4],canMsg.data[5],canMsg.data[6], dan canMsg.data[7]. Masing masing panjangnya satu byte. Ini sesuai dengan struktur dari CAN standar yang mempunyai data 0 … 8 btyes. Data kelembapan diletakkan pada canMsg.data[1]. Sementara byte byte yang lain diisi dengan $\boldsymbol{0x00}$. Bagian void loop() ini kemudian diakhiri dengan sebuah pernyataan yang berhubungan dengan MCP2515 untuk mengirimkan pesan CAN.

Hasil Perangkat Keras

Untuk hasil hardware (perangkat keras) nya, bagian bagian yang sudah digabungkan ditunjukkan di Gambar 13. Bagian bagain tersebut adalah node temperature node, node kelembapan, node receiver, dua sensor temperature & kelembapan DHT11, sebuah LCD I2C, tiga modul MCP2515 & TJA1050 (satu modul untuk tiap tiap node), dan dua resistor untuk terminasi bus sebesar 120 Ω.



Gambar 13. Hasil Perangkat Keras Penggabungan Ketiga Node

Terlihat pada Gambar 13 di atas, dua buah node pada ujung bus CAN adalah node temperature node dan node receiver. Untuk node kelembapan ditempatkan di tengah tengah *bus line*, mengambil data dari bagian tengah line tersebut. Terlihat bahwa dua resistor terminasi 120 Ω menunjukkan bahwa sistem ini mengimplementasikan skema terminasi parallel. Teriminasi Parallel adalah salah satu skema terminasi yang paling banyak diguanakan saat ini. Terminasi Parallel memanfaatkan sebuah resistor yang dipasang di antara *differential lines*, dipasang pada ujung (receiver) dari *transmission line* untuk mengeliminasi refleksi [15].

Sementara Tabel 1 di bawah ini menunjukkan hasil ketika pasangan pesan *id* (satu *id* untuk pesan temperature dan satu *id* lagi untuk pesan kelembapan) divariasikan beberapa kali.

Tabel 1. Hasil tampilan LCD untuk berbagai pasangan nomor *id* pesan

No	Hasil Display LCD	No. *id* pesan kelembapan	No. *id* pesan temperatur
1	Hum (82%)	$id = 0 \times 01$	$id = 0 \times FF$
2	Hum (82%)	$id = 0 \times 11$	$id = 0 \times EE$
3	Hum (82%)	$id = 0 \times 22$	$id = 0 \times DD$
4	Hum (82%)	$id = 0 \times 33$	$id = 0 \times CC$

5	Hum (82%)	$id = 0 \times 44$	$id = 0 \times BB$
6	Hum (82%)	$id = 0 \times 55$	$id = 0 \times AA$
7	Hum (82%)	$id = 0 \times 66$	$id = 0 \times 99$
8	Hum (82%)	$id = 0 \times 77$	$id = 0 \times 88$
9	Hum (82%)	$id = 0 \times 43$	$id = 0 \times 46$
10	Temp (28℃)	$id = 0 \times 39$	$id = 0 \times 36$
11	Temp (28℃)	$id = 0 \times 88$	$id = 0 \times 77$
12	Temp (28℃)	$id = 0 \times 99$	$id = 0 \times 77$
13	Temp (28℃)	$id = 0 \times AA$	$id = 0 \times 66$
14	Temp (28℃)	$id = 0 \times BB$	$id = 0 \times 55$
15	Temp (28℃)	$id = 0 \times CC$	$id = 0 \times 44$
16	Temp (28℃)	$id = 0 \times DD$	$id = 0 \times 33$
17	Temp (28℃)	$id = 0 \times EE$	$id = 0 \times 22$
18	Temp (28℃)	$id = 0 \times FF$	$id = 0 \times 11$
19	Temp (28℃)	$id = 0 \times FE$	$id = 0 \times 12$

Ketika sistem ini dinyalakan untuk pertama kalinya, LCD pertama kali akan menampilkan *welcome message* yang sudah ditulis di dalam program untuk node receiver. Selanjutnya, LCD akan menampilkan nilai temperature atau nilai kelembapan relative yang sudah diukur oleh DHT sensor, tergantung dari angka message *id* masing masing. Tercatat di tampilan yang ditunjukkan oleh LCD tersebut, nilai kelembapan relative adalah sebesar 82% dan nilai temperature adalah sebesar 28℃.

Ketika pada waktu yang berbeda nilai kelembapan relatif dan nilai temperature dimonitor dengan sistem ini dan kemudian hasil pengukuran nya dibandingkan dengan angka yang tercatat pada layar komputer, didapatkan hasil hasil pada Tabel 2 berikut ini.

Tabel 2. Besarnya nilai temperatur dan kelembapan relative tercatat oleh sistem dibandingkan dengan tercatat di layar komputer

No	Parameter	Pengukuran Sistem CAN	Pengukuran Tercatat di Layar Komputer
1	Temperature	25℃	24℃
2	Kelembapan Relatif	80%	74%

Pada Table 2 di atas ditunjukkan, untuk pengukuran temperature, sistem CAN ini mencatat pengukuran $\mathbf{25}$℃, sementara nilai sebesar $\mathbf{24}$℃ tercatat pada layar komputer. Dan untuk pengukuran kelembapan relatif, sistem ini mencatat pengukuran $\mathbf{80}\%$ sementara di layer computer tercatat $\mathbf{74}\%$. Jadi, ada perbedaan sekitar $\mathbf{4}\%$ dan $\mathbf{7.5}\%$ untuk hasil pengukuran temperature dan hasil pengukuran kelembapan relatif antara pencatatan dengan sistem ini dan yang tetera pada layar computer.

Kembali lagi ke Table 1, bisa dilihat bahwa LCD selalu menampilkan pesan pesan dengan *identification number* yang lebih rendah. Misalnya, ketika pesan kelembapan mempunyai $\boldsymbol{Id = 0 \times 001}$ ($\boldsymbol{Id = 000\ 0000\ 0001}$) dan pesan temperature mempunyai $\boldsymbol{Id = 0 \times 0FF}$ ($\boldsymbol{Id = 000\ 1111\ 1111}$), LCD akan menampilkan data kelembapan (dengan besaran $\mathbf{82}\%$). Sebaliknya, ketika pesan kelembapan mempunyai $\boldsymbol{Id = 0 \times 0FF}$ ($\boldsymbol{Id = 000\ 1111\ 1111}$), dan pesan temperature mempunyai $\boldsymbol{Id = 0 \times 011}$ ($\boldsymbol{Id = 000\ 0001\ 0001}$), LCD akan menampilkan data temperature (dengan besaran $\mathbf{28}$℃). Jadi, semakin rendah identifier binary number suatu pesan, semakin tinggi prioritas nya. Bahkan, pada saat terdapat dua identification numbers yang saling berdekatan (misalnya, di Table 1 terdapat pesan kelembapan dengan $\boldsymbol{Id = 0 \times 039}$ ($\boldsymbol{Id = 000\ 0011\ 1001}$), dan pesan temperature dengan $\boldsymbol{Id = 0 \times 036}$ ($\boldsymbol{Id = 000\ 0011\ 0110}$), pesan temperature dengan identifier number nya yang lebih rendah masih lebih prioritas. Kesimpulannya, pesan pesan dengan bit identifier terakhir nol (dominant) akan selalu ditampilkan. Bisa dilihat dan dibandingkan pada empat bit terakhir $\mathbf{1001}$ pada pesan kelembapan dengan empat bit terakhir $\mathbf{0110}$ pada pesan temperature.

Tetapi, ketika sebuah delay ditambahkan pada bagian akhir program transmitter (bisa program transmitter kelembapan atau program transmitter tempetarature), suatu hal berbeda teramati pada tampilan LCD. Yaitu, apabila ditambahkan sebuah pernyataan delay setelah pernyataan untuk mengirimkan pesan, (i.e. yaitu diletakkan sebuah pernyataan delay(1000); setelah baris mcp2515.sendMessage(&canMsg);), maka pesan yang lebih rendah prioritasnya akan kadang kadang ditampilkan oleh LCD. Akan terlihat LCD untuk beberapa saat menampilkan pesan dengan prioritas yang lebih rendah, dan kemudian kembali lagi menampilkan pesan yang lebih prioritas.

Gambar 14 di bawah ini menunjukkan satu contoh LCD yang sedang menampilkan data temperature yang terukur oleh sensor.

Gambar 14. LCD menampilkan data temperature terukur di sekitar sensor DHT11

PEMBAHASAN

Dari hasil pemrograman dan perangkat keras, bisa ditunjukkan bahwa keseluruhan sistem berfungsi dengan baik sesuai dengan rencana dan spesifikasi awal. Penampil LCD berhasil menampilkan secara kontinyu nilai kelembapan relative ($terukur$ 82%) dan nilai temperatur ($terukur$ 28℃) sesuai dengan prioritas pesan. Pesan dengan *identification number* rendah menjadi pesan yang lebih prioritas di jaringan ini. Ketika pertama kali sistem ini dinyalakan, setelah pesan selamat datang, LCD selalu menampilkan pesan yang lebih rendah *identification number* nya.

Pada intinya, hasil hasil eksperimen yang didapatkan telah menunjukan bahwa sensor suhu dan kelembapan relative DHT11 telah berhasil mengukur suhu dan kelembapan relative, dan data suhu serta kelembapan tersebut telah berhasil dikirim ke receiver melewati kontroler CAN, transceiver CAN, dan bus CAN. Secara keseluruhan, sistem juga terbukti stabil dan bisa diandalkan.

Untuk pengembangan ke depan nya, sistem monitor suhu dan kelembapan relative ini bisa saja dimodifikasi sedemikian rupa sehingga jumlah node transmisi menjadi lebih dari dua node. Dengan kata lain, sejumlah titik pengukuran baru bisa ditambahkan di sepanjang bus CAN ini. Variabel yang diukur bisa dibuat seragam. Misalnya, semua titik pengukuran berupa pengukuran temperature atau semua titik pengukuran berupa pengukuran kelembapan. Jadi, di sini berupa pengukuran variable tunggal dengan banyak titik pengukuran. Jadi, sensor sensor dengan tipe yang sama, yang diletakkan di beberapa titik di sepanjang bus, digunakan sebagai cara untuk menyediakan *distributed sensing* dari sebuah variable terukur tunggal.

Opsi yang lain bisa saja misalnya, beberapa pengukuran dibuat sebagai pengukuran temperatur, beberapa lain nya dibuat sebagai pengukuran kelembapan, dan sisa nya dibuat sebagai pengukuran tekanan atau aliran. Jadi, kali ini terdapat beberapa parameter yang diukur, dengan banyak titik pengukuran. Jadi, terdapat pendistribusian sejumlah *discrete sensors* yang berbeda jenis untuk mengukur beberapa variable yang berbeda di sepanjang bus.

Berkaitan dengan pesan nya, pada kedua opsi di atas akan terdapat lebih dari dua pesan dengan pesan *id* nya masing masing. Misalnya, apabila terdapat 20 titik pengukuran, berarti akan terdapat 20 pesan

dengan ***id*** pesan nya masing masing.

KESIMPULAN

Dari eksperimen yang sudah dilakukan, dapat disimpulkan bahwa sistem secara keseluruhan berfungsi sesuai spesifikasi. Node temperature dapat mengirimkan data suhunya ke sepanjang bus. Node kelembapan juga dapat mengirimkan data kelembapan nya ke sepanjang bus. Node receiver kemudian bisa menampilkan data di atas pada LCD.

Pada intinya, bus CAN sebenarnya adalah sebuah jaringan *fieldbus* yang dapat menghubungkan banyak perangkat lapangan ke dalam sebuah jaringan pabrik *(plant network)*. Penelitian ini sudah berhasil memanfaatkan bus CAN ini untuk mendesain suatu sistem pemonitoran temperature dan kelembapan relative yang terdiri dari beberapa node (titik pengukuran). Hanya dengan menggunakan sepasang kabel, perangkat perangkat lapangan yang biasanya digunakan pada aplikasi proses kontrol bisa dihubungkan bersama dan bisa dibuat untuk bisa berkomunikasi satu sama lain. Hal ini tentu saja sangat menguntungkan dengan melihat bahwa sistem ini sederhana dan murah karena semua node berkomunikasi melalui sebuah sistem CAN tunggal, tidak melalui sambungan satu per satu antar node – sehingga mengurangi error, berat, pengkabelan, dan biaya. Keuntungan lain dari sistem bus CAN ini, adalah bahwa *system designer* dapat menentukan pilihan terhadap prioritas pesan. Beberapa designer mungkin sepakat bahwa suatu pesan tertentu lebih signifikan, sehingga diberikan prioritas. Misalnya, sebuah manufaktur motor drives bisa menetapkan bahwa pesan 0010 adalah sebuah sinyal *winding current feedback* dari sebuah motor pada sebuah jaringan CAN, dan 0011 adalah pesan kecepatan tachometer. Karena 0010 merupakan *binary identifier* yang lebih rendah, maka pesan pesan yang berkaitan dengan besarnya nilai *winding current* akan selalu lebih diprioritaskan pada bus dibandingkan pesan pesan yang berkaitan dengan pembacaan tachometer. Dan terakhir, yang paling penting adalah bahwa sistem yang dibuat ini stabil dan bisa diandalkan.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Kemenristek Dikti dan Lembaga Penelitian dan Pengabdian kepada Masyarakat (LPPM) Universitas Jenderal Soedirman atas pembiayaan penelitian ini melalui skema Riset Peningkatan Kompetensi (RPK) tahun 2023.

DAFTAR PUSTAKA

Curtis D. Johnson. 2000. *Process Control Instrumentation Technology*. Sixth Edition. Columbus, Ohio: Prentice Hall.

J. da Silva Sa, J. J. da Silva, M. G. Wanzeller and J. S. da Rocha Neto. 2005. Monitoring of temperature using smart sensors based on CAN architecture. 15th International Conference on Electronics, Communications and Computers (CONIELECOMP'05): 218-222.

Xu Yan, Guo Tao, Zhu Jie and Chen Wei. 2011. Based on single-chip microcomputer temperature and humidity data acquisition system design. Proceedings of 2011 International Conference on Electronics and Optoelectronics: V2-310-V2-313.

Q. Zhu, D. Zhu and X. Su. 2010. Distributed remote temperature monitoring and acquisition system based on CAN bus. Prognostics and System Health Management Conference: 1-4.

L. Zhang, L. Sun and W. Lu. 2010. A Temperature Monitoring System of Power Cable Joints Based on the Combining of CAN Wired Transmission and ZigBee Wireless Network. 2nd International Conference on Information Engineering and Computer Science: 1-4.

Franklyn W Kirk, Thomas A. Weedon, Philip Kirk. 2010. *Instrumentation*. Fifth Edition. USA: American Technical Publishers Inc.

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

Wolfhard Lawrenz. 2013. *CAN System Engineering – From Theory to Practical Applications*. Second Edition. Wolfenbuttel, Germany: Springer.

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

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

Arduino CC. 2023. *Arduino® Nano*. Arduino CC. Product Reference Manual. Number: SKUA000005.

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

Phillips Semiconductors. 2003. *TJA1050 High Speed CAN Transceiver*. Phillips Semiconductors. Product Specification.

Mouser Electronics. DHT11 Humidity & Temperature Sensor. Mouser Electronics. Data Sheet.

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