LAPORAN
SISTEM MIKROPROSESSOR

SPI (*Serial Peripheral Interface*)

Disusun oleh

Nurul Khairiyah Amiruddin	04161054
Wildan Liulinnuha Ichsan	04161078

INSTITUT TEKNOLOGI KALIMANTAN
BALIKPAPAN
2018

DAFTAR ISI

BAB I

LANDASAN TEORI

1.1 USB

Universal Serial Bus (USB) adalah standar bus serial untuk perangkat penghubung, biasanya digunakan pada komputer namun juga dapat digunakan di peralatan lainnya seperti konsol permainan, ponsel dan PDA.

Sistem USB mempunyai desain yang asimetris, yang terdiri dari pengontrol *host* dan beberapa peralatan terhubung yang berbentuk "pohon" dengan menggunakan peralatan hub yang khusus [3].

Desain USB ditujukan untuk menghilangkan perlunya penambahan *expansion card* ke ISA komputer atau bus PCI, dan memperbaiki kemampuan *plug-and-play* (pasang-dan-mainkan) dengan memperbolehkan peralatan - peralatan ditukar atau ditambah ke sistem tanpa perlu me-*reboot* komputer. Ketika USB dipasang, ia langsung dikenal sistem komputer dan memproses *device driver* yang diperlukan untuk menjalankannya [2].

USB dapat menghubungkan peralatan tambahan komputer seperti *mouse*, *keyboard*, pemindai gambar, kamera digital, *printer*, *hard disk*, dan komponen *networking*. USB kini telah menjadi standar bagi peralatan multimedia seperti pemindai gambar dan kamera digital [3].

Gambar 1.1 USB Arduino

1.2 Arduino

Arduino adalah pengendali mikro *single-board* yang bersifat *open-source*, diturunkan dari *wiring platform*, dirancang untuk memudahkan penggunaan elektronik dalam berbagai bidang. *Hardware*nya memiliki prosesor Atmel AVR dan *software*nya memiliki bahasa pemrograman sendiri [4].

Arduino juga merupakan *platform hardware* terbuka yang ditujukan kepada siapa saja yang ingin membuat purwarupa peralatan elektronik interaktif berdasarkan *hardware* dan *software* yang fleksibel dan mudah digunakan. Mikrokontroler diprogram menggunakan bahasa pemrograman arduino yang memiliki kemiripan *syntax* dengan bahasa pemrograman C. Karena sifatnya yang terbuka maka siapa saja dapat mengunduh skema *hardware* arduino dan membangunnya .

Arduino menggunakan keluarga mikrokontroler ATMega yang dirilis oleh Atmel sebagai basis, namun ada individu/perusahaan yang membuat *clone* arduino dengan menggunakan mikrokontroler lain dan tetap kompatibel dengan arduino pada *level hardware*. Untuk fleksibilitas, program dimasukkan melalui *bootloader* meskipun ada opsi untuk mem-*bypass* bootloader dan menggunakan *downloader* untuk memprogram mikrokontroler secara langsung melalui port ISP.

Seperti Mikrokontroler yang banyak jenisnya, Arduino lahir dan berkembang, kemudian muncul dengan berbagai jenis. Diantaranya yang paling sering dipakai antara lain:

a. Arduino Uno

Jenis yang ini adalah yang paling banyak digunakan. Terutama untuk pemula sangat disarankan untuk menggunakan Arduino Uno. Banyak sekali referensi yang membahas Arduino Uno. Versi yang terakhir adalah Arduino Uno R3 (Revisi 3), menggunakan ATMEGA328 sebagai Mikrokontrolernya, memiliki 14 pin I/O digital dan 6 pin *input analog*. Untuk pemprograman cukup menggunakan koneksi USB *type* A to *type* B. Sama seperti yang digunakan pada USB *printer*.

Gambar 1.2 Arduino UNO

b. Arduino Mega

Mirip dengan Arduino Uno, sama-sama menggunakan USB *type* A to B untuk pemprogramannya. Tetapi Arduino Mega, menggunakan *chip* yang lebih tinggi ATMega2560. Dan tentu saja untuk pin I/O Digital dan pin *input analog*-nya lebih banyak dari Uno.

c. Arduino Leonardo

Bisa dibilang Leonardo adalah saudara kembar dari Uno. Dari mulai jumlah pin I/O digital dan pin *input analog*nya sama. Hanya pada Leonardo menggunakan Micro USB untuk pemprogramannya.

d. Arduino Nano

Sepertinya namanya, Nano yang berukulan kecil dan sangat sederhana ini, menyimpan banyak fasilitas. Sudah dilengkapi dengan FTDI untuk pemograman lewat Micro USB. 14 Pin I/O Digital, dan 8 Pin *input* Analog (lebih banyak dari Uno). Dan ada yang menggunakan ATMEGA168, atau ATMEGA328.

e. Arduino Mini

Fasilitasnya sama dengan yang dimiliki Nano. Hanya tidak dilengkapi dengan Micro USB untuk pemograman. Dan ukurannya hanya 30 mm x 18 mm saja. [1]

1.3 *Interface* SPI

SPI memiliki beberapa bagian *interface* di dalamnya baik *interface* yang populer maupun yang telah lalu. Untuk saat ini, bus SPI menetapkan empat sinyal logika:

a. SCLK: *Serial Clock* (keluaran dari *master*)
b. MOSI: *Master Out Slave In* (*output* data dari *master*)
c. MISO: *Master In Slave Out* (*output* data dari *slave*)
d. SS: *Slave Select* (sering aktif ketika *low*, *output* dari *master*)

Sementara nama pin di atas adalah yang paling populer, di masa lalu pin alternatif konvensi penamaan kadang-kadang digunakan, dan jadi nama pin *port*

SPI untuk produk IC yang lebih tua dan mungkin berbeda dari yang digambarkan dalam ilustrasi pada bab ini, terdapat beberapa penamaan lain pada pin SPI antara lain :

a. SCLK: SCK

   Output Master → *Input Slave* (MOSI):

b. SIMO, MTSR - sesuai dengan MOSI pada perangkat *master* dan *slave*, menghubungkan satu sama lain.

c. SDI, DI, DIN, SI - pada perangkat pendukung; terhubung ke MOSI di *master*, atau ke koneksi di bawah.

d. SDO, DO, DOUT, SO - pada perangkat *master*; terhubung ke MOSI pada *slave*, atau ke koneksi di atas.

   Input Master ← *Output Slave* (MISO):

e. SOMI, MRST - sesuai dengan MISO pada perangkat *master* dan *slave*, menghubungkan satu sama lain.

f. SDO, DO, DOUT, SO - pada perangkat *slave*, terhubung ke MISO di *master*, atau ke koneksi di bawah.

g. SDI, DI, DIN, SI - pada perangkat utama, menghubungkan ke MISO pada *slave*, atau koneksi di atas.

h. *Slave Select*:

   SS: $\overline{\text{SS}}$, SSEL, CS, $\overline{\text{CS}}$, CE, nSS, / SS, SS #

   Dengan kata lain, MOSI (atau SDO pada *master*) terhubung ke MOSI (atau SDI pada *slave*). MISO (atau SDI pada *master*) menghubungkan ke MISO (atau SDO pada *slave*). *Slave Select* memiliki fungsi yang sama dengan *chip select* dan digunakan sebagai pengganti konsep pengalamatan. Nama pin selalu dikapitalisasi seperti pada *Slave Select*, *Serial Clock*, dan *Master Output Slave Input*.

1.4 SPI *(Serial Peripheral Interface)*

Serial Peripheral Interface (SPI) adalah suatu spesifikasi komunikasi serial sinkron yang digunakan untuk komunikasi jarak pendek, terutama dalam sistem *embedded*. *Interface* pertama dikembangkan oleh Motorola pada pertengahan 1980-an dan telah menjadi standar *de facto*.

Perangkat SPI berkomunikasi dalam mode *full duplex* serta menggunakan arsitektur *master-slave* dengan *master* tunggal. Perangkat *master* dapat melakukan perintah untuk membaca dan menulis. Kemudian beberapa perangkat slave didukung melalui pemilihan dengan slave select (SS).

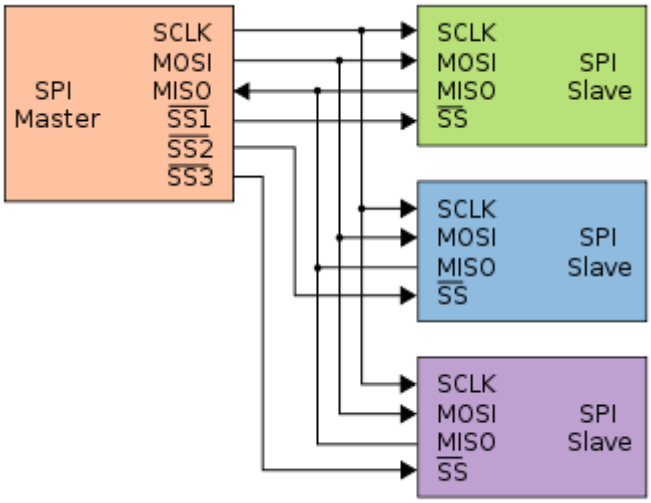
Terkadang SPI disebut juga sebagai bus serial empat pin, yang tidak juah berbeda dengan bus serial tiga, dua, atau satu pin. SPI dapat secara akurat digambarkan sebagai komunikasi antarmuka serial sinkron, tetapi berbeda dari protokol *Synchronous Serial Interface* (SSI), yang juga merupakan protokol komunikasi serial sinkron empat pin. Protokol SSI menggunakan pensinyalan differensial dan hanya menyediakan satu saluran komunikasi simpleks tunggal.

Gambar 1.3 Komunikasi SPI

SPI atau *Serial Peripheral Interface* juga merupakan komunikasi seri *synchronous* yang berarti harus menggunakan *clock* yang sama untuk mengsinkronisasi deteksi bit pada *receiver*. Biasanya hanya digunakan untuk komunikasi jarak pendek dengan mikrokontroler lain yang terletak pada papan rangkaian yang sama. Bus SPI dikembangkan untuk menyediakan komunikasi dengan kecepatan tinggi dengan menggunakan pin mikrokontroler yang sedikit. SPI melibatkan *master* dan *slave.* Keduanya mengirimkan dan menerima data secara terus menurus, namun *master* bertanggung jawab untuk menyediakan sinyal *clock* untuk *transfer* data. Gambar 3 menunjukkan komunikasi antara *master* dan *slave* pada komunikasi SPI. *Master* menyediakan *clock* dan data 8 bit pada pin *master-out-slave-in* (MOSI) dimana data tersebut di-*transfer* satu bit per pulsa *clock* menuju pin MOSI pada *slave*. Delapan bit data juga diberikan dari *slave* ke *master* melalui pin *master in slave* out menuju pin MISO pada *master*. Biasanya pin $\overline{SS}$ (*slave select*) diberi ground (active *low*) untuk menjadikannya sebagai *slave* [5].

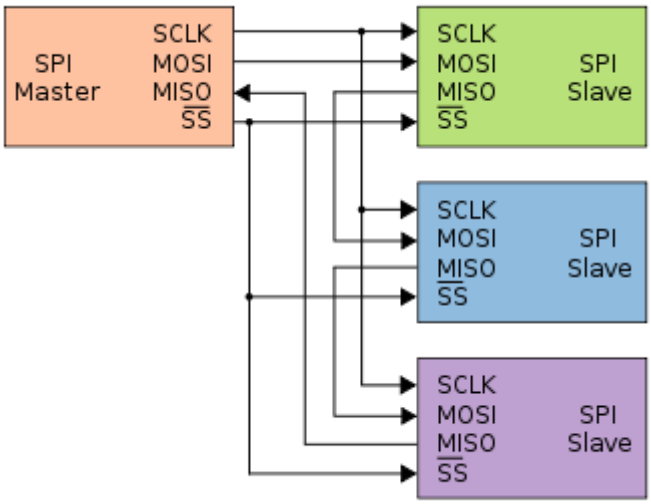
Pada aplikasi dari SPI pada *slave* yang lebih dari satu, SPI memiliki konfigurasi yang mengatur penggunaan serta pengambilan atau pengiriman data dari *master* dan *slave*. Konfigurasi tersebut antara lain:

a. *Independent Slave Configuration*



Gambar 1.4 SPI konfigurasi *independent slave*

Dalam konfigurasi *slave* independen, terdapat garis pilih secara independen untuk setiap *slave*. Sebuah *resistor pull-up* antara sumber daya dan garis pilih *slave* sangat disarankan untuk setiap perangkat independen untuk mengurangi *cross-talk* antar perangkat. Ini adalah cara SPI biasanya digunakan. Karena pin MISO dari *slave* terhubung bersama, mereka diminta untuk menjadi pin *tri-state* (tinggi, rendah atau impedansi tinggi). [6]



Gambar 1.5 SPI konfigurasi *daisy chain*

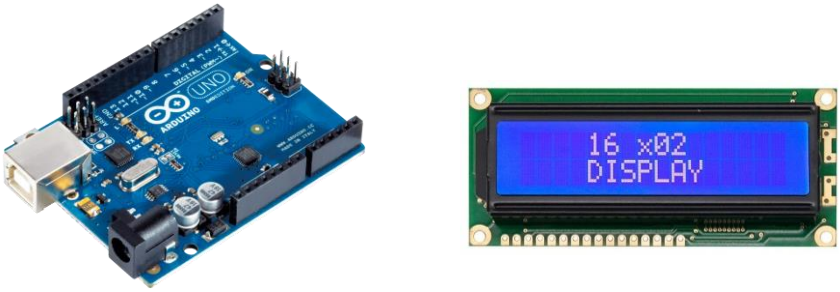
Beberapa perangkat yang mengimplementasikan SPI dapat dihubungkan dalam konfigurasi *daisy chain*, *output slave* pertama dihubungkan ke *input slave* kedua, dan lain - lain. Seluruh rantai bertindak sebagai *shift register* komunikasi, *daisy chaining* sering dilakukan dengan register geser untuk menyediakan *bank input* atau *output* melalui SPI. Setiap *slave* menyalin *input* ke *output* dalam siklus *clock* berikutnya sampai SS line aktif *low* menjadi *high*. Fitur seperti itu hanya membutuhkan satu jalur SS dari *master*, daripada jalur SS terpisah untuk setiap *slave*.

Aplikasi lain yang dapat berinteraksi secara potensial dengan SPI yang memerlukan konfigurasi *daisy chain* termasuk SGPIO, JTAG, [5] dan *Two Wire Interface*. [7]

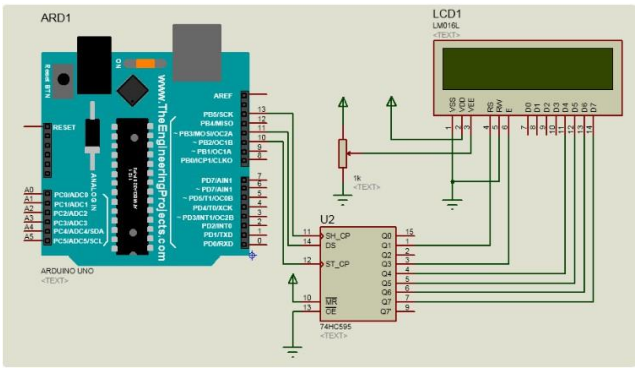
BAB II
PEMBAHASAN

2.1 Hardware

Hardware dari SPI (*Serial Peripheral Interface*) yang digunakan adalah arduino UNO dan LCD. Pada komunikasi ini Arduino atau mikrokontroller berfungsi sebagai *master* dan LCD sebagai *slave.*



Gambar 2.1 Arduino dan LCD

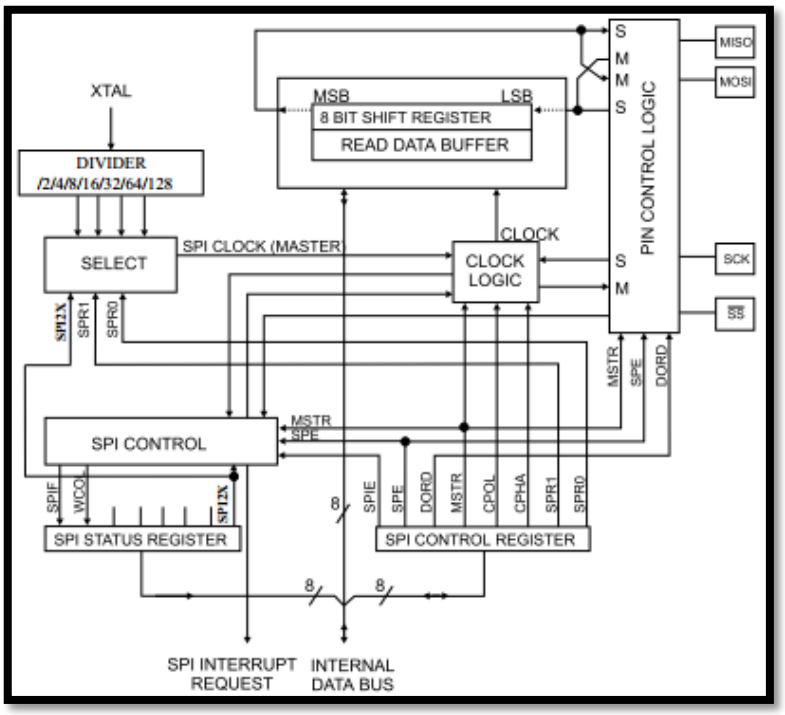


Gambar 2.2 Simulasi SPI Menggunakan Proteus

Empat jalur yang membangun komunikasi SPI pada Arduino Uno terletak pada pin 10 (SS), 11 (MOSI), 12 (MISO) dan 13 (SCK). Selain keempat pin tersebut, papan Arduino juga menyediakan deretan pin (kecuali SS) yang dikenal sebagai ICSP (*In-Circuit Serial Programming*). Selain SCK, MISO dan MOSI, *header* ICSP juga dilengkapi dengan pin 5V, Ground, dan Reset.

2.1.1 *Input* dan *Output*

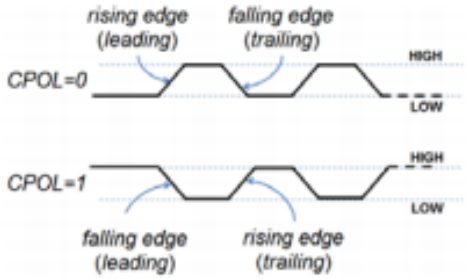
SPI menggunakan dua pin untuk transfer data, yaitu SDI (Din) dan SDO (Dout). Bus SPI memiliki sebuah SCLK (Shift Clock), yang berfungsi sebagai jalur penyedia detak untuk sinkronisasi data. SPI memiliki satu atau lebih CE (Chip Enable) untuk memilih *slave* mana yang akan berkomunikkasi dengan *master*. Pada beberapa perangkat, keempat pin ini (SDI, SDO, SCLK dan CE) juga dikenal sebagai MOSI (*Master* Out *Slave* In), MISO (*Master* In *Slave* Out), SCK, dan SS (*Slave Select*). Arsitektur SPI dapat dijelaskan dari blok diagram di bawah ini



Gambar 2.3 Blok Diagram SPI

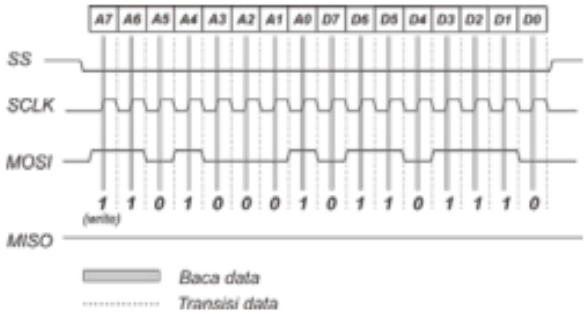
SPI terdiri dari dua buah *shift* register yang terletak di bagian *master* dan di bagian *slave*. Pembangkit detak (*clock generator*) terdapat di bagian *master*, dan berfungsi memberi detak untuk *shift* register pada *master* atau *slave.* Kedua *shift* register memliki lebar 8-bit.Sehingga, setelah 8-detak, maka isi dari kedua *shift* register akan saling dipertukarkan. Jika *master* ingin mengirim 1 -byte (8-bit) data, maka *master* akan meletakaan data pada *shift* register. Setelah 8 detak, maka isi dari *shift* register pada *master* akan sampai pada *slave*. Sebaliknya, jika *master* ingin menerima data, maka *slave* akan menempatkan data pada *shift* register. Setelah 8 detak, maka isi dari *slave* akan diterima oleh *master*.

Pada SPI, polaritas detak dikenal sebagai CPOL (*Clock Polarity*) dan CPHA (*Clock Phase*). Jika CPOL = 0, basis nilai dari detak adalah 0 (*LOW*), sedangkan jika CPOL = 1, basis nilai dari detak adalah 1 (HIGH). Jika CPHA = 0, sampling akan dilakukan pada transisi pertama (*leading*) dari detak, sedangkan jika CPHA = 1, sampling akan dilakukan pada transisi kedua (*trailing*) dari detak. Ilustrasi polaritas ini ditunjukkan pada Gambar di bawah ini. Jika CPOL = 0, maka *leading* dari detak adalah *rising edge*, sedangkan jika CPOL = 1, maka *leading* dari detak adalah *falling edge*.



Gambar 2.4 Polaritas dan Transisi Detak

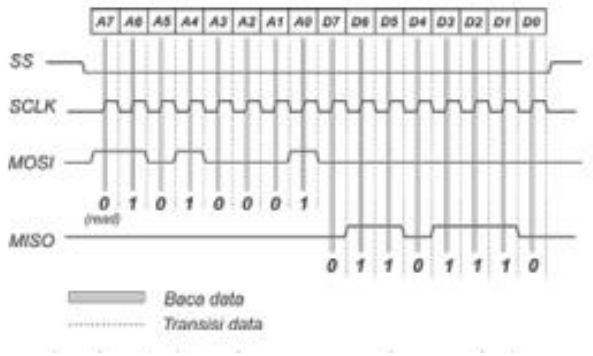
Pengiriman data dilakukan dengan mengirim alamat terlebih dahulu, kemudian secara langsung diikuti oleh data. Untuk membedakan antara menulis (*write*) dan membaca (*read*), maka kita harus melakukan pengaturan pada bit MSB (A7) dari alamat. Jika MSB = 1, proses tersebut adalah menulis, sedangkan jika MSB = 0, proses tersebut adalah membaca. Selanjutnya, sisa 7- bit selanjutnya akan digunakan untuk menandai alamat yang dituju. Untuk menulis 1 -byte data melalui SPI, maka langkah pertama yang harus dilakukan adalah membuat **SS** menjadi **0** (*LOW*). Asumsi *master* ingin menulis ke alamat **1010001**. Karena menulis, MSB alamat akan dibuat bernilai 1 (HIGH). Dengan demikian, keseluruhan 8-bit alamat adalah **11010001**. Setiap bit pada alamat ini akan tergeser setiap 1 detak.



Gambar 2.5 Proses Menulis pada SPI

Pada Gambar di atas , grafik transisi data diperlihatkan. Pada gambar tersebut, *master* mengirimkan data ke *slave* yang beralamat di 1010001 dan data yang dikirim adalah 01101110.

Untuk kegiatan membaca yang dilakukan oleh *master*, langkah pertama yang harus dilakukan adalah membuat SS menjadi *LOW*. Segera setelah SS menjadi *LOW*, maka sistem akan mengetahui bahwa komunikasi akan segera dibangun. *Master* akan meletakkan 8-bit alamat yang akan dibaca. Karena membaca, bit **MSB** (A7) akan dibuat **nol**. Alamat akan dikirim ke *slave* melalui MOSI (*Master* Out *Slave* In). Mengetahui alamat yang dituju, maka *slave* akan menyiapkan datanya pada shift register, dan akan dikirim ke *master* melalui MISO (*Master* In *Slave* Out). Ingat, bahwa pergeseran tiap bit data akan mengikuti detak yang tersedia. Proses membaca diakhiri dengan membuat SS kembali menjadi HIGH.



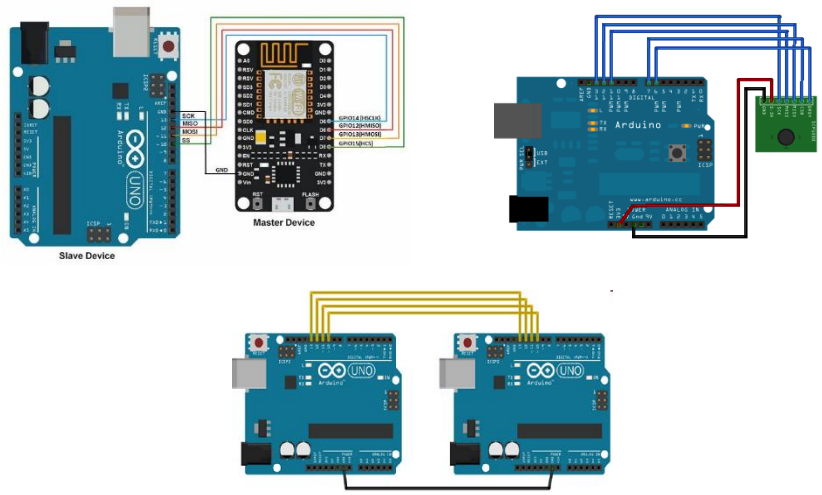
Gambar 2.6 Proses Membaca pada SPI

Pada gambar tersebut, terlihat bahwa alamat yang ingin dibaca pada lokasi *slave* adalah **1010001**. Pada lokasi tersebut, ternyata data yang tersedia adalah

01101110. Dengan demikian, data inilah yang dipersiapkan oleh *slave* untuk dibaca oleh *master*.

2.1.2 Implementasi

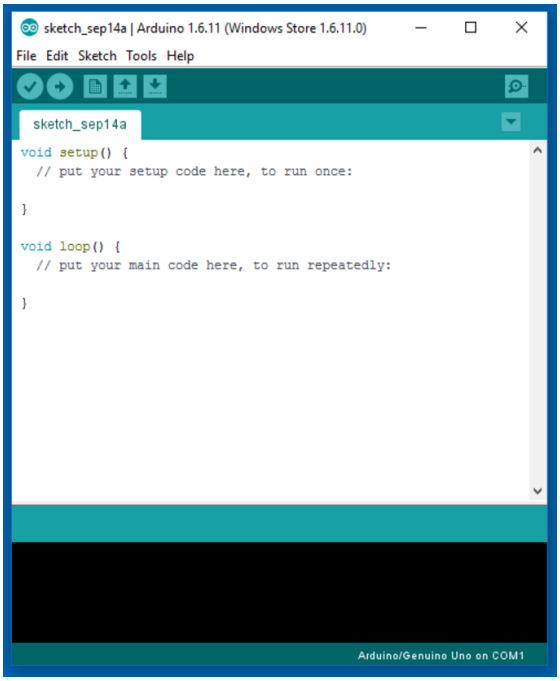
SPI (*Serial Peripheral Interface)* merupakan sistem komunikasi yang dapat dilakukan antara dua perangkat peripheral atau pun lebih. Namun, yang dapat bertindak sebagai *master* hanyalah mikrokontroler. Selain contoh yang telah dijelaskan sebelumnya yaitu antara mikrokontroller Arduino dengan LCD, sistem komunikasi SPI dpat dilakukan antara mikrokontroler dengan perangkat peripheral lain seperti dengan sensor, actuator, atau bahkan dengan arduino. Pengaplikasian SPI dapat dilihat pada gambar di bawah ini



Gambar 2.7 Implementasi SPI antara Mikrokontroller dengan Perangkat Lain

2.2 Software

Software dari SPI (*Serial Peripheral Interface*) ini salah satu contohnya adalah Arduino IDE



Gambar 2.8 Software Arduino IDE

Kode Program Arduino biasa disebut sketch dan dibuat menggunakan bahasa pemrograman C. Program atau sketch yang sudah selesai ditulis di Arduino IDE bisa langsung dicompile dan diupload ke Arduino Board. Secara sederhana, sketch dalam Arduino dikelompokkan menjadi 3 blok (lihat gambar di atas) yaitu *Header, Setup dan Loop.*

Header merupakan bagian yang biasanya ditulis definisi-definisi penting yang akan digunakan selanjutnya dalam program, misalnya penggunaan library dan pendefinisian variable. *Setup* merupakan awal program Arduino berjalan, yaitu di saat awal, atau ketika *power on* Arduino *board.* Biasanya di blok ini diisi penentuan apakah suatu pin digunakan sebagai *input* atau *output*, menggunakan perintah pinMode. Selain itu, initialisasi *variable* juga bisa dilakukan di blok ini. *Loop* merupakan blok yang akan dieksekusi secara terus menerus. Apabila program sudah sampai akhir blok, maka akan dilanjutkan dengan mengulang eksekusi dari awal blok. Program akan berhenti apabila tombol *power* Arduino di matikan.

2.2.1 Inisialisasi

Inisialisasi yang dilakukan untuk sistem komunikasi antar perangkat atau SPI adalah seperti yang dijelaskan sebelumnya yaitu antara Arduino sebagai *master* dan LCD sebagai *slave*. Dapat dilihat pada gambar di bawah ini

Gambar 2.9 Implementasi SPI antar Arduino dengan LCD

Untuk dapat menajalankan program diperlukan *library* dari SPI itu sendiri. Selanjutnya, menentukan bahwa Arduino sebagai *master* dan LCD sebagai *slave.* Penentuan *master* dan *slave* tersebut berdasarkan kode yang telah kita buat. Selanjutnya, program akan di *compile* dan di*upload* pada hardware Arduino atau dapat disimulasikan dengan Proteus. Jika telah di*upload*, maka akan berlangsung proses komunikasi antara Arduino dengan LCD.

BAB 3
PENUTUP

3.1 Kesimpulan

Komunikasi secara SPI merupakan protokol data serial sinkron yang digunakan oleh mikrokontroler untuk berkomunikasi dengan satu atau lebih perangkat periferal Komunikasi SPI menjawab kekurangan komunikasi secara asinkron yang merupakan komunikasi yang umum terjadi melalui TX dan RX pada mikrokontroler. Pada pembahasan ini, sistem komunikasi secara SPI dilakukan dengan menggunakan *software* Arduino IDE dan Proteus. Digunakan perangkat keras berupa mikrokontroller Arduino sebagai *master* dengan LCD sebagai *slave*. Berdasarkan simulasi yang dilakukan sistem komunikasi dapat berjalan dengan baik tanpa adanya *error.*

Daftar Pustaka

[1] *Arduino.cc* (diakses 30 Oktober 2018)

[2] Masya, Fajar. *Pengembangan IoT dengan arduino,* Media Komputindo, Jakarta, 2016.

[3] Nugroho, Adi. *Pemrograman sederhana arduino,* Graha Ilmu, Jakarta, 2008.

[4] Santoso, Hari. *Arduino untuk Pemula,* Elang Sakti. Trenggalek. 2015

[5] SPI Block Guide V03.06.Motorola Inc. Illinois. 2003

[6] *www.dorkbotpdx.org* (diakses 30 Oktober 2018)

[7] *www.maximintegrated.com* (diakses 30 Oktober 2018)