TEKNIK ANTARMUKA SECARA SERIAL PERIPHERAL INTERFACE (SPI) MENGGUNAKAN PLATFORM ARDUINO DAN MATLAB

I Nyoman Kusuma Wardana
Jurusan Teknik Elektro, Politeknik Negeri Bali, Bukit Jimbaran, Badung ,Bali – 80364 Telp. (0361) 701981
email: kusumawardana@pnb.ac.id

Abstrak : Pada port serial mikrokontroler, komunikasi antara Transmitter (TX) dan Receiver (RX) dikenal sebagai komunikasi secara asinkron (*asynchronous*). Operasi pada mikrokontroler umumnya berdasarkan detak yang disinkronkan menggunakan kristal (*clock*) tunggal, maka perbedaan kecepatan dapat menjadi masalah ketika dua perangkat yang sama-sama dilengkapi dengan sumber detak yang berbeda saling berkomunikasi. Ini adalah salah satu kelemahan komunikasi secara asinkron. Salah satu solusi yang digunakan untuk mengatasi masalah ini adalah dengan menggunakan komunikasi secara Serial Peripheral Interface (SPI) yang bersifat sinkron. Pada penelitian ini, antarmuka SPI pada sistem multi mikrokontroler dengan menggunakan platform Arduino dan MATLAB telah dipaparkan. Bedasarkan hasil penelitian, sistem komunikasi secara SPI yang terkontrol MATLAB telah berjalan, dan komunikasi master-slave berjalan baik.

Kata kunci : *SPI, mikrokontroler, MATLAB, Arduino, sinkron, asinkron, protokol komunikasi*

Interface Technique with Serial Peripheral Interface (SPI) Using Platform Arduino and Matlab

Abtract **:** *The serial communication type that commonly used on the TX and RX of the microcontroller pins is asynchronous. Since computers normally rely on everything being synchronized to a single clock, this can be a problem when two systems with slightly different clocks try to communicate with each other. To overcome this problem, the Serial Peripheral Interface (SPI) that works on synchronous mode is proposed. In this research, the SPI interface of multi microcontrollers is explained. It is obtained that the master-slave communication has been built properly.*

Keywords*: SPI, microcontroller, MATLAB, Arduino, synchronous, asynchronous, communication protocol*

1. PENDAHULUAN

Teknik antarmuka memegang peranan penting pada mikrokontroler. Teknik ini dapat mencakup antarmuka mikrokontroler dengan sensor, mikrokontroler dengan aktuator, mikrokontroler dengan komputer, mikrokontroler dengan perangkat jaringan, atau antarmikrokontroler sendiri. Secara umum, komunikasi dibangun dengan dua jenis, yaitu komunikasi secara serial dan secara paralel.

Antarmuka mikrokontroler dengan perangkat eksternal saat ini didominasi oleh sistem komunikasi secara serial. Pada mikorkontroler ATmega328 (mikrokontroler yang digunakan pada Arduino Uno), komunikasi secara serial dapat berupa Serial Universal Synchronous and Asynchronous Serial Receiver and Transmitter (USART), Two-wire Interface (TWI), dan Serial Peripheral Interface (SPI) [1].

Pada port serial mikrokontroler, komunikasi antara Transmitter (TX) dan Receiver (RX) dikenal sebagai komunikasi secara asinkron (*asynchronous*), tidak secara sinkron (*synchronous*) [1],[2]. Sistem ini dikatakan asinkron sebab tidak terdapat kontrol ketika data dikirim atau tidak ada jaminan bahwa kedua perangkat yang berkomunikasi memiliki kecepatan detak yang sama [2],[3].

Operasi pada mikrokontroler umumnya berdasarkan detak yang disinkronkan menggunakan kristal (*clock*) tunggal, maka perbedaan kecepatan dapat menjadi masalah ketika dua perangkat yang sama-sama dilengkapi dengan sumber detak yang berbeda saling berkomunikasi. Sebagai contoh, pada tulisan ini, beberapa mikrokontroler yang masing-masing memiliki sumber detak yang berbeda akan dikomunikasikan.

Untuk mengatasi hal ini, komunikasi secara asinkron akan menggunakan bit tambahan berupa start bit dan stop bit di setiap paket data yang dikirim. Kedua perangkat yang berokomunikasi harus memiliki sistem konfigurasi yang sama. Pada sistem komunikasi umumnya data dikirim dalam banyak paket, maka kecepatan akan berkurang, sebab terdapat bit ekstra yang harus dikirim. Demikian pula, jika kedua perangkat yang berkomunikasi memiliki kecepatan sampling yang berbeda, sistem komunikasi akan gagal atau salah dalam menerjemahkan [2].

Salah satu solusi yang digunakan untuk mengatasi masalah ini adalah dengan menggunakan komunikasi secara *Serial Peripheral Interface* (SPI). SPI adalah protokol komunikasi secara *synchronous* antara dua perangkat (*master* dan *slave*), yang memisahkan antara jalur data dan jalur detak [1],[2],[3],[4].

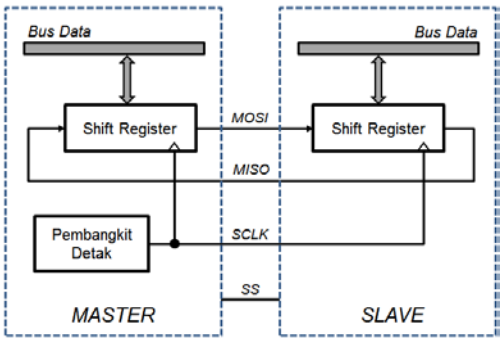
SPI dimulai oleh Motorola (sekarang Freescale), dan diterapkan secara luas oleh berbagai perusahaan semikonduktor. ISP merupakan komunikasi serial *full-duplex*, yang memungkinkan komunikasi dua arah antara master dan slave secara simultan. Dalam penerapannya, SPI banyak digunakan pada komunikasi antarperagkat, seperti EEPROM, ADC, DAC, sensor dan aktuator lainnya [5].

Tujuan penelitian ini membangun komunikasi multi-platform menggunakan platform Arduino dengan memanfaatkan komunikasi secara SPI. Jenis papan Arduino yang digunakan adalah Uno, yang merupakan papan Arduino yang paling banyak digunakan [6]. Pengaturan utama komunikasi akan menggunakan perangkat lunak MATLAB. MATLAB adalah bahasa komputasi teknis yang paling popular, dan banyak digunakan pada sistem pemodelan, simulasi, dan grafis [7]. MATLAB juga dapat digunakan untuk membangun *Graphical User Interface* (GUI) sebagai antarmuka pengguna [8].

2. TINJAUAN PUSTAKA

2.1 Arsitektur Komunikasi SPI

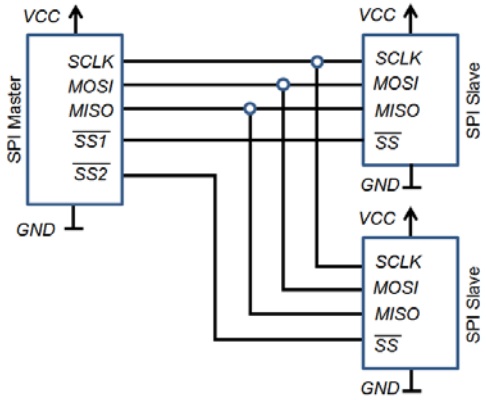
SPI menggunakan dua pin untuk transfer data, yaitu SDI (Din) dan SDO (Dout). Bus SPI memiliki sebuah SCLK (Shift Clock), yang berfungsi sebagai jalur penyedia detak untuk sinkronisasi data. Terakhir, SPI memiliki satu atau lebih CE (Chip Enable) untuk memilih slave mana yang akan berkomunikkasi dengan 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 diperlihatkan pada Gambar 1.



Gambar 1. Arsitektur 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, baik untuk bagian master maupun bagian slave.

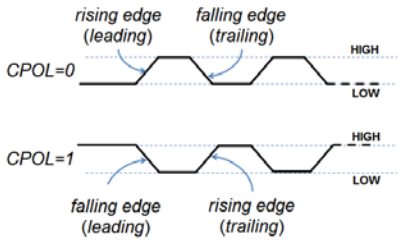
Kedua *shift* register memliki lebar 8-bit. Dengan demikian, setelah 8-detak, maka isi dari kedua *shift* register akan saling dipertukarkan. Jika master ingin mengirim 1-byte (8-bit) data, maka master akan meletakaan data pada *shift* register. Setelah 8 detak, maka isi dari *shift* register pada master 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. Contoh konfigurasi antara master dengan dua buah slave ditunjukkan pada Gambar 2.



Gambar 2. Konfigurasi Master dengan dua buah Slave [5]

Master bertugas membangkitkan detak. Terkait dengan penggunaan Arduino, yang bertugas sebagai master adalah Arduino itu sendiri, sedangkan *slave* adalah berbagai sensor, aktuator, atau bahkan Arduino lainnya. Tergantung banyaknya *slave*, pin SS yang dibutuhkan oleh master juga bervariasi.

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 3. Perhatikan bahwa jika CPOL = 0, maka *leading* dari detak adalah *rising edge*, sedangkan jika CPOL = 1, maka *leading* dari detak adalah *falling edge*.

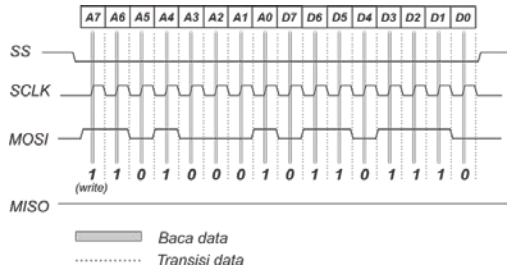


Gambar 3. Polaritas dan transisi detak

2.2 Menulis dan Membaca pada SPI

Dalam komunikasi SPI, skema 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.

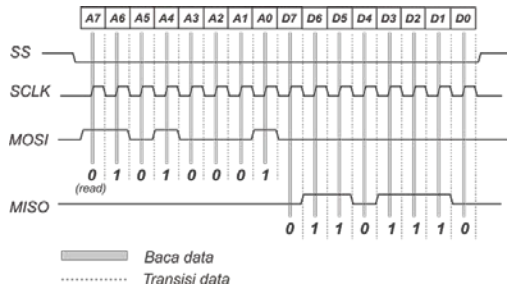
Untuk menulis 1-byte data melalui SPI, maka langkah pertama yang harus dilakukan adalah membuat **SS** menjadi **0** (LOW). Asumsi master ingin menulis ke alamat **1010001**. Karena menulis, MSB alamat akan dibuat bernilai 1 (HIGH). Dengan demikian, keseluruhan 8-bit alamat adalah **11010001**. Setiap bit pada alamat ini akan tergeser setiap 1 detak.



Gambar 4. Contoh proses menulis pada SPI

Pada Gambar 4, 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.

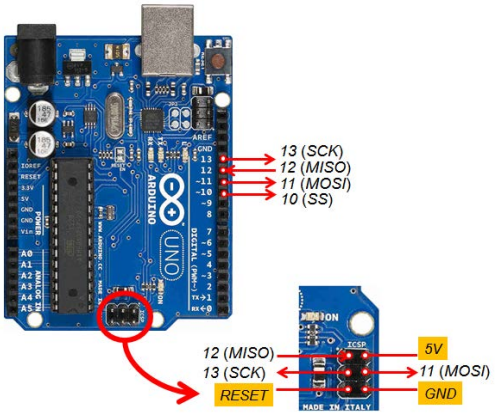


Gambar 4. Contoh proses membaca pada SPI

Contoh pembacaan data oleh master diperlihatkan pada Gambar. 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.

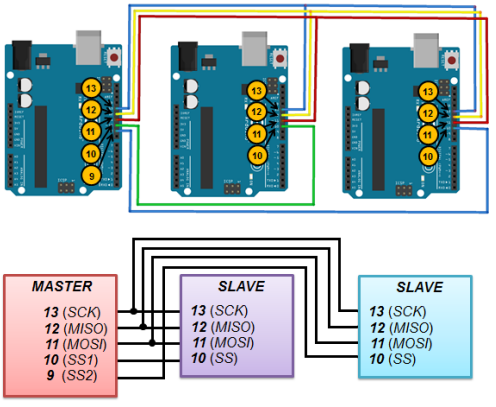
2.3 SPI pada Arduino

Keempat 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. Posisi pin-pin ini diperlihatkan pada Gambar 5.



Gambar 5. Pin pada Arduino Uno untuk komunikasi secara SPI

Pada Gambar 6, tiap-tiap pin yang bersesuaian akan saling terhubung. *Slave* dapat ditambah atau dikurangi. Penggunaan kabel untuk **SS** (*Slave Select*) dapat bertambah, sesuai dengan jumlah *slave*. Satu buah slave akan ditangani oleh satu buah SS. Dengan demikian, diperlukan sebanyak ***N* + 3** pin untuk master, dimana ***N*** adalah jumlah slave yang digunakan.

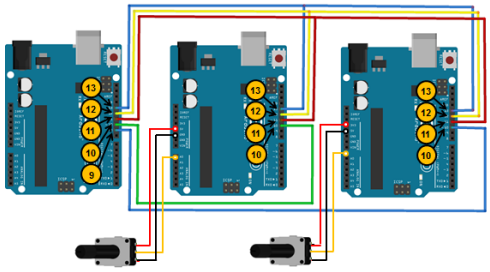


Gambar 5. Skema satu master dua slave

3. METODE PENELITIAN

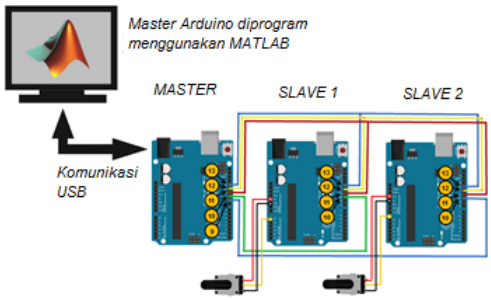
3.1 Rancangan Umum Penelitian

Pada penelitian ini, tiga buah Arduino akan dikonfigurasi untuk berkomunikasi secara SPI. Dua buah potensiometer ditempatkan pada pin A0 di tiap-tiap *slave*. sebuah LED ditempatkan di pin digital 7 (pemasangan LED tidak diperlihatkan pada Gambar 6) *Slave* 1 akan dikontrol menggunakan pin SS 10 pada master, sedangkan *Slave* 2 akan dikontrol melalui pin SS 9 pada master.



Gambar 6. Skema konfigurasi penelitian

Pada penelitian ini, pemrograman SPI yang dikontrol melalui MATLAB untuk dua buah papan Arduino telah dilakukan. Pemrograman SPI akan memanfaatkan sebuah Arduino sebagai master, dan dua Arduino lainnya sebagai slave. Arduino yang bertindak sebagai master akan diprogram dengan MATLAB, sedangkan slave akan diprogram menggunakan IDE Arduino.



Gambar 6. Skema pembagian pemrograman

4. HASIL DAN PEMBAHASAN

4.1 Pemrograman Arduino untuk Slave

Pada mikrokontroler Atmel (jenis mikrokontroler pada Arduino Uno), register untuk melakukan konfigurasi komunikasi secara SPI dikenal sebagai SPCR (SPI Control Register). SPCR adalah register 8-bit, dan konfigurasi komunikasi SPI kita akan bergantung dari nilai yang dituliskan pada SPCR.. Pada penelitian ini, komunikasi SPI diaktifkan dengan mengeset bit SPE, dan mengaktifkan interupsi untuk SPI (mengeset bit SPIE). Cara yang dapat digunakan untuk mengeset kedua bit ini adalah sebagai berikut: `SPCR = (1<<SPIE)|(1<<SPE)`. Pemrograman sederhana ini ditunjukkan pada Gambar 7.

```
        byte dataKu;
byte buffer;
boolean prosesSensor = false;
boolean prosesLED = false;
boolean ledState = false;

const int ledPin =  7;
const int PinSS = 10;
const int PinMOSI = 11;
const int PinMISO = 12;
const int PinSCK = 13;

void setup (void)
{
  SPCR = 0;
  SPCR = (1<<SPIE)|(1<<SPE);
  pinMode(PinMISO, OUTPUT);
  pinMode(PinMOSI, INPUT);
  pinMode(PinSS, INPUT);
  pinMode(PinSCK, INPUT);
  pinMode(ledPin, OUTPUT);

}

// Routine untuk interupsi
ISR (SPI_STC_vect)
{
  buffer = SPDR;
    if (buffer == 100)
    {
      prosesLED = true;
      ledState = false;
    }
    else if (buffer == 101)
    {
      prosesLED = true;
      ledState = true;
    }
    else if (buffer == 1)
    {
      prosesSensor = true;
    }
    else if (buffer == 0)
    {
      prosesSensor = false;
    }
    else
    {
    }
}

void loop (void)
{
  if(prosesSensor == true){
    dataKu = analogRead(A0)/4;
    SPDR =  dataKu;
    prosesSensor = true;
  }
  if(prosesLED == true){
    if (ledState == true){
      digitalWrite(ledPin, HIGH);
    }
    else if (ledState == false){
      digitalWrite(ledPin, LOW);
    }
  prosesLED = false;
  }
delay(50);
}
```

Gambar 7. Skrip pemrograman pada slave

Pada program Arduino Gambar 7, digunakan sebuah rutin interupsi pada SPI, yaitu **SPI_STC_vect**. Jika rutin ini aktif, maka program mengecek data yang diterima pada **SPDR** (SPI Data Register). Jika yang diterima adalah **100**, mikrokontroler akan memadamkan LED, sedangkan jika diterima **101**, mikrokontroler akan menyalakan LED. Jika SPDR bernilai **1**, mikrokontroler akan membaca nilai potensiometer, dan jika bernilai **0**, pembacaan sensor berhenti. Nilai-nilai ini dapat ditentukan sesuai dengan keinginan pengguna.

4.2 Pemrograman MATLAB untuk Master

MATLAB dan Arduino berkomunikasi menggunakan komunikasi secara serial. MATLAB telah mengembangkan suatu pustaka untuk membangun komunikasi dengan Arduino, termasuk di dalamnya memasukkan komunikasi secara SPI. Instalasi pustaka ini dilakukan secara daring melalui menu pada perangkat lunak MATLAB. Pengguna dapat secara eksplisit memasukkan nomer port COM yang digunakan dan menulis pustaka I2C pada MATLAB. Gambar 8 menunjukkan proses membangun koneksi. Pada Gambar 8, nomer port yang digunakan adalah 33 dan pustaka SPI secara eksplisit digunakan.

```
>> a = arduino('com33','uno',...
              'libraries','SPI')

a =
  arduino with properties:

                    Port: 'COM33'
                   Board: 'Uno'
     AvailableAnalogPins: [0,1,2,3,4,5]
    AvailableDigitalPins:
[2,3,4,5,6,7,8,9,10,11,12,13]
                Libraries: {'SPI'}
```

Gambar 8. Membangun koneksi dengan Master Arduino melalui MATLAB

Proses selanjutnya adalah membuat objek untuk slave 1 dan slave 2. Slave 1 memiliki menggunakan SS di pin 10, dan slave 2 memiliki menggunakan SS di pin 9, seperti yang ditunjukkan pada Gambar 9. Semakin banyak slave yang digunakan, maka semakin banyak pin SS yang dibutuhkan.

```
>> SPIdevice1 = spidev(a, 10);
>> SPIdevice2 = spidev(a, 9);
```

Gambar 9. Membuat objek untuk kedua slave

4.3 Membaca dan Menulis Data ke *Slave*

Membaca dan Menulis ke *Slave* dilakukan seluruhnya melalui MATLAB. Sebagai contoh, jika ingin menyalakan dan memadamkan LED di pin 7 pada *slave* pertama dan kedua, pengguna dapat menggunakan sintaks `writeRead`, seperti pada Gambar 10.

```
>> %nyalakan LED di Slave 1
>> writeRead(SPIdevice1, 101, 'uint8');

>> %nyalakan LED di Slave 2
>> writeRead(SPIdevice2, 101, 'uint8');

>> %memadamkan LED di Slave 1
>> writeRead(SPIdevice1, 100, 'uint8');

>> %memadamkan LED di Slave 2
>> writeRead(SPIdevice2, 100, 'uint8');
```

Gambar 10. Mengontrol LED untuk kedua slave

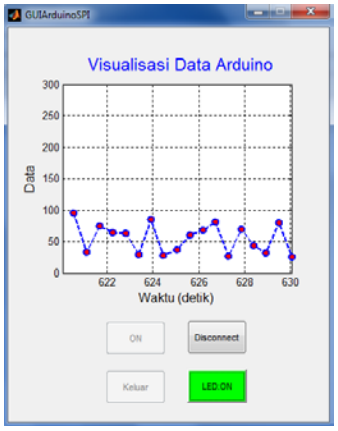
Selain kegiatan menulis (*write*) ke *slave*, pengguna juga dapat membaca (*read*) data dari *slave*. Berdasarkan Gambar 6, penelitian ini telah menyertakan dua buah potensiometer pada pin A0 untuk kedua slave. Potensiometer tersebut dapat diputar-putar, dan sebuah variabel untuk menampung nilai sensor dapat dideklarasikan pada MATLAB, seperti yang ditunjukkan pada Gambar 11. Data potensiometer di *slave* 1 dan 2 dapat diperoleh dengan menggunakan fungsi yang sama dengan sintaks fungsi menulis, yaitu `writeRead`. Yang membedakan antara menulis dan membaca adalah jenis data yang dikirim ke Arduino, dan data yang berbeda ini telah diprogram sebelumnya di slave Arduino, sesuai dengan program yang ditampilkan pada Gambar 7.

```
>> data = writeRead(SPIdevice1, 1,
'uint8')

data1 =
  168

>> data2 = writeRead(SPIdevice2, 1,
'uint8')

data2 =
  56
```

Gambar 11. Membaca nilai potensiometer untuk kedua slave

4.4 Pengembangan Antarmuka MATLAB

MATLAB adalah bahasa komputasi teknis yang popular digunakan oleh kalangan akademik, peneliti, maupun industri [7]. Dengan demikian, dengan memahami pemrograman MATLAB, kita dapat menambah kemampuan aplikasi berbasis mikrokontroler, misalnya dengan menambahkan antarmuka berupa *Graphical User Interface* (GUI), basis data, pelaporan (*reporting*), dan sebagainya. Pada penelitian ini, dibuat sebuah antarmuka untuk komunikasi master dan sebuah slave. Contoh antarmuka ditampilkan pada Gambar 12.



Gambar 12. Contoh antarmuka menggunakan MATLAB untuk sebuah slave

5 SIMPULAN DAN SARAN

5.1 Simpulan

Komunikasi secara SPI menjawab kekurangan komunikasi secara asinkron. Komunikasi asinkron adalah komunikasi yang umum terjadi pada melalui TX dan RX pada mikrokontroler. Pada penelitian ini, sistem komunikasi secara SPI yang terkontrol MATLAB telah dilakukan, dan sistem dapat berjalan

dengan baik. MATLAB memiliki potensi yang besar untuk dikembangkan lebih lanjut, terutama dari sisi pemrosesan program yang bersifat kompleks.

5.2 Saran

MATLAB adalah bahasa komputasi teknis yang multi fungsi, sedangkan Arduino adalah platform berbasis mikrokontroler yang *user-friendly*. Kombinasi kedua platform ini dapat menghasilkan aplikasi yang beragam dan kompleks. Penelitian selanjutnya dapat menerapkan penggunaan dapat melibatkan penggunaan database, pelaporan, sistem daring, dan sebagainya.

DAFTAR PUSTAKA

[1] Anonim, 2013, *Official User Manual: 8-bit Microcontroller with 4/8/16/32/K Bytes In-System Programmable Flash*, Atmel Corporation.

[2] Grusin, M., 2010, *Serial Peripheral Interface (SPI)*, A Tutorial by Sparkfun. Tersedia di https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi (diakses 7 Desember 2016)

[3] Blum, M., 2013, *Exploring Arduino: Tools and Techniques for Engineering Wizardry*, John Wiley & Sons, Inc.

[4] Anand, N., Joseph, G., Oomen, S.S., 2014, *Design and Implementation of a High Speed Serial Peripheral Interface*, IEEE International Conference on Advances in Electrical Engineering.

[5] Wardana, I N.K., 2016, *Teknik Antarmuka MATLAB dan Arduino*, Vaikuntha International Publication

[6] Anonim, *Arduino UNO & Genuino UNO*, laman resmi Arduino (www.arduino.cc)

[7] Anonim, 2016, *MATLAB® Primer*. The MathWorks, Inc.

[8] Inavov, V., Inavov, S., Brojboiu, M., 2008, *MATLAB Graphical User Interface for System with Dallas Microcontroller*. IEEE International Symposium on Power Electronics, Electrical Drives, Automation and Motion.