## II.3 Teori Pendukung

Beberapa teori yang mendasari pelaksaan penelitian proyek akhir yang berkaitan dengan perangkat keras dan metode-metode dari sistem yang direalisasikan.

### II.3.1 Ubi Jalar

**Gambar II.1 Ubi Jalar (Sumber: food.detik.com)**

**Ubi jalar** (*Ipomoea batatas* L.) atau dalam bahasa Inggrisnya ***sweet potato*** adalah sejenis tanaman budidaya. Bagian yang dimanfaatkan adalah akarnya yang membentuk umbi dengan kadar gizi (karbohidrat) yang tinggi [7].

Ubi jalar di Indonesia mempunyai potensi pengembangan yang prospektif sebagai bahan baku industri, baik untuk industri pangan maupun non-pangan. Sebagai bahan pangan sumber karbohidrat dan komoditi pangan penting di Indonesia selain beras dan ubi jalar di budidayakan di seluruh Indonesia dan menjadi komoditi pasar yang di usahakan penduduk Indonesia mulai dari daerah dataran rendah sampai dataran tinggi. Keberhasilan pengolahan ubi jalar yang setara dengan beras dan dapat mempercepat upaya diversifikasi pangan, sehingga dapat mengurangi ketergantungan pada beras [8].

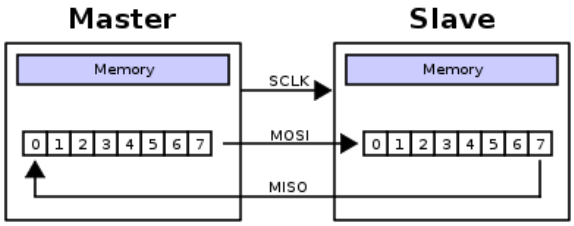
### II.3.2 Suhu

Suhu adalah derajat panas atau dinginnya suatu benda atau lingkungan. Semakin tinggi suhu suatu benda maka suhu nya semakin panas begitupun sebaliknya semakin rendah suhu suatu benda maka semakin dingin suhunya. Dalam hal ini suhu yang menjadi batas penelitian penulis adalah suhu pada air mendidih

dan suhu makanan. Pada skala celcius, 0 °C adalah titik di mana air membeku dan 100 °C adalah titik didih air pada tekanan 1 atmosfer [9] .

### II.3.3 *Serial Peripheral Interface* (SPI)

*Serial Peripheral Interface* (SPI) merupakan salah satu protokol komunikasi yang digunakan pada proses pengiriman data antara perangkat mikrokontroler dengan perangkat kecil misalnya shift register dan sensor. SPI merupakan model komunikasi synchronous yaitu menggunakan jalur yang terpisah untuk data dan clock sehingga kedua perangkat dapat di sinkronkan secara sempurna. Untuk memulai komunikasi, pada bus master dilakukan konfigurasi *clock*, dengan catatan frekuensi atau kecepatan pengiriman data antara SPI pada perangkat *master* serta perangkat *slave* harus sama. *Master* akan memilih perangkat *slave* dengan mengeluarkan logika 0, lalu *master* akan menunggu proses yang telah dijadwalkan oleh *master* itu sendiri seperti urutan interupsi, konversi analog *to* digital/ADC dan lain-lain. Kemudian saat periode itu telah selesai maka *master* mengirimkan *clock* yang merupakan tanda untuk memulai sebuah proses komunikasi serial. *Setup hardware* komunikasi SPI diperlihatkan pada gambar II.2.



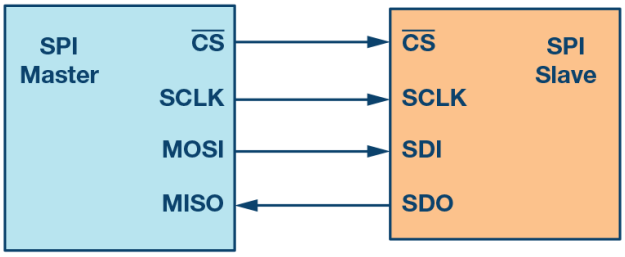

**Gambar II.2 Setup Hardware SPI (Sumber: Wikipedia.org)**

Interface Bus SPI Master dan slave terhubung oleh empat jalur. Pada tiap jalurnya memiliki informasi dan membawa sinyal tertentu yang didefinisikan oleh protokol dari bus SPI. Keempat jalur tersebut adalah:

- MOSI (*Master Output Slave Input*), ini merupakan sinyal output dari perangkat *master* yang merupakan *shift register* dari *master* menuju input dari *slave*.
- MISO (*Master Input Slave Output*), ini merupakan sinyal input dari perangkat *master* untuk menerima data *shift register* dari perangkat *slave* menuju *master*.

- SCK atau SCLK (*Serial Clock*), ini merupakan *clock* yang dihasilkan *master* yang berguna mengadakan komuniaksi SPI dan untuk melakukan *shifting* terhadap *shift register* dari kedua perangkat.
- SS' (*Slave Select*), merupakan pin yang digunakan untuk memilih *slave* mana yang akan diajak berkomunikasi oleh *master* (dengan asumsi lebih dari satu *slave*).

Sinyal MOSI, SCK, dan SS merupakan sinyal yang berasal dari bagian *master* untuk dikirim menuju bagian *slave*. Sedangkan MISO digunakan sebagai penerima sinyal dari bagian *slave*. Berikut ini merupakan diagram *interface* antara perangkat master dan perangkat *slave* yang diperlihatkan pada gambar II.3.




**Gambar II.3 Diagram Interface Single Master SPI (Sumber: analog.com)**

Dengan demikian, setiap *clock* SPI yang melakukan tranmisi *full duplex* akan mengalami:

- *Master* mengirimkan satu bit menuju *slave*, kemudian *slave* membacanya dalam satu *line* yang identik.
- *Slave* mengirimkan satu bit menuju *master*, kemudian *master* juga membacanya dalam satu *line* yang identik.

### II.3.3 UART (Universal Asynchronous Receiver Transmitter)

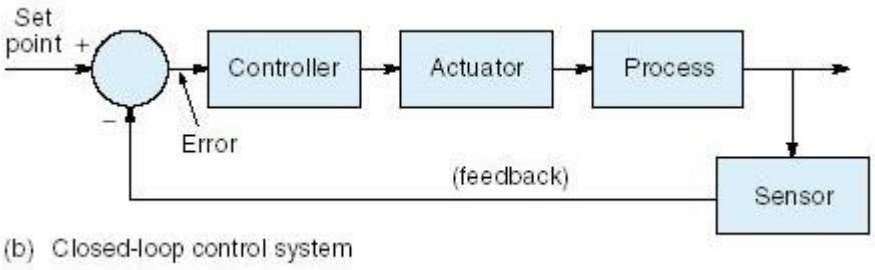
UART (*Universal Asynchronous Receiver Transmitter)* merupakan salah satu protokol yang sering digunakan pada proses komunikasi data serial antara perangkat satu dengan lainnya seperti, proses komunikasi antara dua mikrokontroler. Pada proses pengiriman data yang dilakukan, *clock* pada perangkat pengirim dan *clock* pada perangkat penerima harus sama, karena paket data yang akan dikirimkan tiap bit-nya bergantung pada *clock*. Karena hal tersebut maka model *asynchronous* ini memiliki keunggulan dalam pengiriman data sebab hanya

menggunakan satu kabel transmisi. Sebaliknya berbeda dengan model synchronous pada protokol seperti I2C (*Inter-Integrated Circuit*) dan SPI (*Serial Peripheral Interface*), kedua protokol tersebut membutuhkan paling sedikit dua kabel pada proses transmisi datanya, yaitu transmisi data serta transmisi *clock*. Kelemahan dari model *asynchronous* ada pada jarak dan kecepatan transmisinya, karena semakin jauh jarak serta semakin cepat kecepatan transmisinya maka paket – paket bit dari data yang dikirimkan menjadi terdistorsi hal ini menyebabkan baik data yang dikirim maupun data yang diterima bisa mengalami *error*. Selain itu *error* juga bisa terjadi ketika paket bit data yang tidak cocok dengan *clock* dari mikrokontrolernya. Paket tersebut bergantung pada besarnya nilai baudrate dengan satuan bit per detik. Pada pengiriman data UART terdapat parameter – parameter yang dapat diatur seperti *start* bit, *parity* bit, dan *stop* bit. Pengaturan ini harus sesuai antara perangkat pengirim dan perangkat penerima apabila tidak sesuai maka data tidak akan bisa diterima dengan baik. Data yang dikirim adalah data 8 bit atau 1 byte. Jika ditambah dengan tiga parameter diatas maka total bit data yang dikirim adalah 11 bit. Dari format data inilah setiap data yang terbaca dapat diterjemahkan menjadi bit-bit yang merepresentasikan data tertentu. *Error* dapat terjadi ketika menggunakan *clock* mikrokontroler untuk nilai tertentu saja. Pada paket data UART, *clock* yang dikirimkan bergantung dari nilai *baudrate*. Karena protokol ini universal, maka *baudrate* yang ada adalah nilai – nilai yang sudah ditentukan dari kisaran nilai 110 sampai demgan 11059200 bps (bit per detik) atau lebih. Semakin cepat *clock* yang digunakan oleh mikrokontroler atau mikroprosesor maka baudrate akan semakin cepat juga. Umumnya *baudrate* yang digunakan dalam pengiriman data antar mikrokontroler adalah 9600 bps.

### II.3.4 Sistem Kendali *On-Off*

Sistem kendali *on-off* atau sering disebut dengan *two-point control* merupakan salah satu teknik kendali yang memiliki 2 kondisi yaitu *on* dan *off*. Metode kendali *on-off* merupakan salah satu metode kendali yang paling umum digunakan karena implementasinya yang mudah. Namun pengendalian *ON-OFF* dengan demikian hanya bisa diterapkan pada sistem yang bereaksi secara relatif

pelan/lambat seperti pada sistem kendali suhu, atau sistem gerak dengan kecepatan lambat. Proses perancangan sistem kendali tertutup ditunjukkan pada Gambar II.2.




**Gambar II.4 Diagram Sistem Kendali On-Off (Sumber: www.kuliah.unpatti.ac.id)**

Perancangan kendali *on-off* seperti Gambar II.4 harus mendefinisikan terlebih dahulu SP (*set point*) dari variabel hasil pengukuran sensor yang diinginkan yaitu CV (*control variable*). Lalu setiap hasil pengukuran sensor akan dibandingkan dengan *set point*, apabila belum memenuhi *set point* maka akan mengubah kondisi aktuator *ON* atau *OFF* sehingga variabel yang dikontrol menjadi sama dengan nilai yang diinginkan.