BAB 2
TINJAUAN PUSTAKA

2.1 Kajian Pustaka

Dalam kajian pustaka penulis memilih beberapa jurnal yang dijadikan sebagai contoh acuan dalam pengembangan tugas akhir yang dibuat.

Pada jurnal "Peramalan Jumlah Permintaan Produksi Menggunakan Metode Jaringan Syaraf Tiruan (JST) *Backpropagation*" merupakan sebuah jurnal yang dibuat oleh Mira Febrina, Faula Arina, Ratna Ekawati. Dalam jurnal tersebut tersaji penelitian menggunakan metode Jaringan Syaraf Tiruan (JST) dengan faktor terkait yaitu hasil penjualan, harga dan stok barang jadi. Pengolahan JST menggunakan *software* MATLAB. Penerapan metode JST di PT.XYZ menggunakan algoritma *backpropagation*. Arsitektur jaringan syaraf tiruan yang digunakan yaitu 3 *input layer*, 1 *output layer*, dan 1 *hidden layer* serta fungsi aktivasi yang digunakan *logsig* dan *purelin*. *Logsig* untuk *hidden layer* dan *purelin* untuk *output layer*. Rancangan arsitektur jaringan syaraf tiruan terbaik untuk peramalan permintaan v-belt AJGG B-65 adalah jaringan *multi layer feedforward* dengan struktur *neuron* 20-1 dengan 1 (satu) *hidden layer*, *learning rate* (lr) yang digunakan 0,1 dan momentum *constant* (mc) 0,2. Nilai *Mean Square Error* (MSE) pelatihan jaringan sebesar 0,001 Nilai MAPE pengujian data sebesar 5,7134%. Hasil peramalan jaringan syaraf tiruan yang telah di denormalisasi yaitu 12142 pcs, 30927 pcs, 27259 pcs, 40259 pcs, 14529 pcs, 23135 pcs, 19611 pcs, 10434 pcs, 6062 pcs, 35201 pcs, 16289 pcs dan 31763 pcs [1].

Selanjutnya, Jurnal "*Internet Of Things* (IOT) Sistem Pengendalian Lampu Menggunakan *Raspberry Pi* Berbasis *Mobile*". Jurnal ini dibuat oleh Yoyon Efendi. Dalam jurnal ini memuat beberapa penelitian seperti pengendalian lampu berbasis IoT, *Internet of thing* (IoT) bisa dimanfaatkan pada gedung untuk mengendalikan peralatan elektronik seperti lampu ruangan yang dapat dioperasikan dari jarak jauh melalui jaringan komputer. Penelitian ini bertujuan untuk membangun perangkat *remote*

control dengan memanfaatkan teknologi internet untuk melakukan proses pengendalian lampu berbasis *mobile*. Penelitian dilakukan dengan membangun sebuah *prototype* dan aplikasi berbasis *mobile* menggunakan bahasa pemrograman *python*. Dalam penelitian ini terdapat fitur kendali yaitu kendali satu lampu yang digunakan untuk menghidupkan satu lampu dan kendali dua digunakan untuk menghidupkan lampu secara bersamaan [2].

Dari kedua jurnal diatas terdapat beberapa teori dan pengimplementasian yang dapat dijadikan sebagai sebuah acuan contoh dalam perancangan tugas akhir. Pada jurnal pertama yang menggunakan teori jaringan syaraf tiruan hasil yang didapat pada jurnal tersebut dapat dijadikan contoh karena sedikit berhubungan dengan tugas akhir yang akan penulis buat, yaitu sebuah alat dengan sistem yang dapat membuat sebuah pola penjadwalan dengan jaringan syaraf tiruan yang nantinya dapat mengoperasikan lampu otomatis sesuai pembentukan pola tersebut. Jurnal selanjutnya berkaitan dengan perancangan *prototype* dan membuat program aplikasi *mobile* dengan bahasa pemrograman *python* menggunakan *Rasberry Pi* 3 sebagai pengendali lampu jarak jauh dengan jaringan internet yang dapat diterapkan pada peralatan elektronik seperti lampu.

2.2 Landasan Teori

Dalam landasan teori, memuat beberapa teori-teori dan penjelasan beberapa komponen yang akan digunakan dalam pembuatan tugas akhir. Seperti hal nya kajian pustaka, beberapa teori yang akan dijelaskan dalam bab berikutnya adalah untuk membantu penulis dalam pembuatan tugas akhir yang berkaitan dengan beberapa teori berikut

2.2.1 Jaringan Syaraf Tiruan

Jaringan syaraf tiruan (JST) dikembangkan berdasarkan proses pembelajaran otak manusia, disebut tiruan karena jaringan syaraf ini diimplementasikan dengan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama pembelajaran . ada beberapa teknik pembelajaran JST, salah satu yang paling sering digunakan adalah *backpropagation* (BP). Inti dari BP adalah melakukan perhitungan

maju untuk mengetahui output dan kinerja jaringan, selanjutnya dilakukan perhitungan mundur untuk mengetahui *error* jaringan yang kemudian digunakan sebagai perubahan bobot. Seiring dengan kebutuhan mendapatkan proses pembelajaran yang lebih cepat, maka dikembangkan beberapa algoritma pembelajaran baru dengan prinsip BP, diantaranya: *gradient descent, resillent backpropagation, quasi newton, dan levenberg marquardt* [6].

2.2.1.1 Proses Normalisasi

Proses normalisasi merupakan proses yang dilakukan untuk merubah data *input* dan target yang akan digunakan dalam proses pelatihan dan pengujian JST berada pada suatu *range* tertentu. Terdapat banyak beberapa cara metode dalam melakukan normalisasi seperti, *min-max*, *decimal scaling, sigmoid*, dan lainnya . Metode normalisasi yang sering digunakan adalah normalisasi min-max dimana nromalisasi ini akan menskalakan suatu nilai yang diberikan kedalam nilai baru antara 0 sampai dengan 1 berdasarkan nilai maksimum dan minimum *datashet* tersebut. berikut *formula* untuk mendapatkan nilai baru dengan menggunakan normalisasi *min-max* [6]. Diberikan nilai yang bersesuaian (Ş), dimana k = 0,1,...n, maka nilai normalisasinya adalah.

$$\acute{S} = \frac{S - min\ (Ş k)}{max(Ş k) - min(Ş k)}$$

keterangan:

$\acute{S}$ = nilai *input* yang telah dinormalisasi

S = nilai *input* yang belum dinormalisasi

$min\ (Ş k)$ = nilai *input* terkecil

$max(Ş k)$ = nilai *input* terbesar

2.2.1.2 Fungsi Aktivasi Sigmoid

Fungsi Sigmoid *biner* memiliki nilai pada *range* 0 sampai 1. Oleh karena itu fungsi ini sering digunakan untuk JST yang membutuhkan nilai *output* pada interval 0 sampai 1. Fungsi sigmoid *biner* dirumuskan sebagai berikut [6]:

$$\boldsymbol{f}(\boldsymbol{x}) = \frac{1}{1 + e^{x}}$$

dengan,

$$\boldsymbol{f}'(x) = x * (1 - x)$$

keterangan:

$\boldsymbol{f}(\boldsymbol{x})$ = nilai aktivasi sebelumnya

$\boldsymbol{f}'(x)$ = nilai aktivasi yang telah ter-*update*

x = nilai *input* yang telah dinormalisasi

2.2.1.3 Pelatihan Data

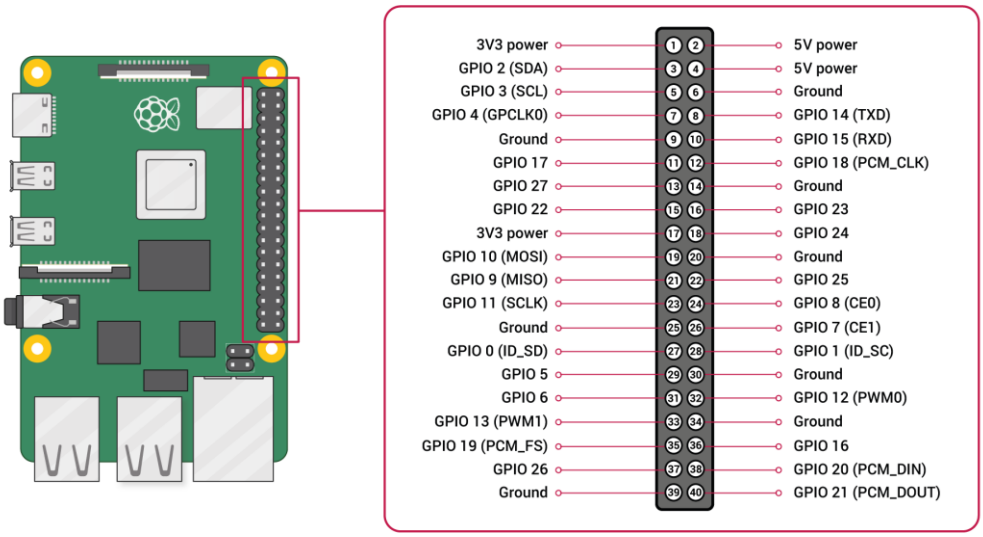
Pelatihan data merupakan *validasi* model jaringan. Pelatihan data dilakukan beberapa kali *trial* dan *error* untuk mendapatkan jaringan terbaik dengan menentukan jumlah neuron [6]. Pelatihan data pada alat sistem rumah pintar dilakukan dengan meberikan *learning rate* 15000, hal ini dilakukan agar hasil *output* yang didapat *error* nya tidak terlalu jauh dari data input yang diolah. Dalam pelatihan data harus mempertimbangkan laju dari iterasi pelatihan agar proses nya tidak terlalu lambat. Bobot hasil pelatihan terbaik rancangan penjadwalan JST disimpan untuk proses pengujian data agar didapatkan hasil uji yang baik juga.

2.2.2 Raspberry PI

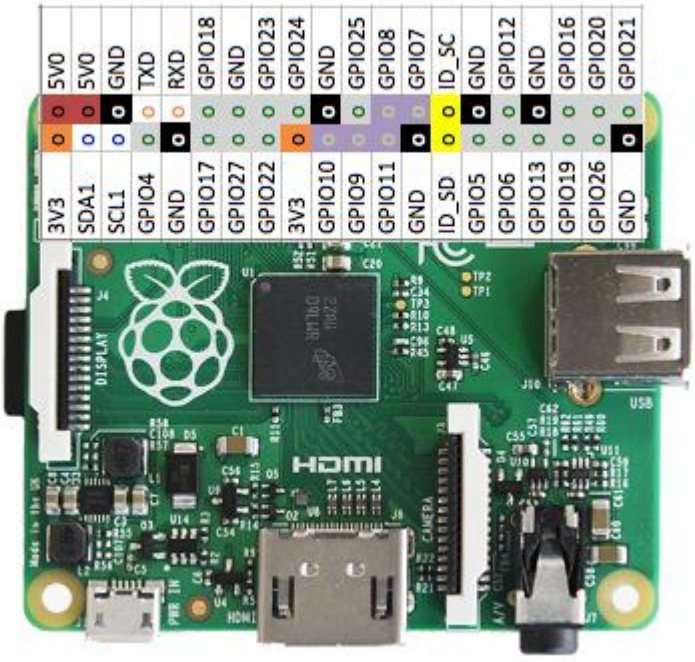
Raspberry Pi, sering juga disingkat dengan nama Raspi, adalah komputer papan tunggal (*Single board Circuit/SBC*) yang memiliki ukuran sebesar kartu kredit. *Raspberry Pi* bisa digunakan untuk berbagai keperluan, seperti *spreadsheet, game*, bahkan bisa digunakan sebagai *media player* karena kemampuannya dalam memutar video *high definition*. *Raspberry Pi* dikembangkan oleh yayasan nirlaba, *Rasberry Pi Foundation* yang digawangi sejumlah *developer* dan ahli komputer dari Universitas Cambridge, Inggris.

Raspberry Pi memiliki dua model yaitu model A dan model B. Secara umum *Raspberry Pi* Model B, 512MB RAM. Perbedaan model A dan B terletak pada memori yang digunakan, Model A menggunakan memori 256 MB dan model B 512 MB. Selain

itu model B juga sudah dilengkapai dengan *ethernet port* (kartu jaringan) yang tidak terdapat di model A. Desain *Raspberry Pi* didasarkan seputar SoC (*System-on-a-chip*) *Broadcom* BCM2835, yang telah menanamkan *prosesor* ARM1176JZF-S dengan 700 MHz, *VideoCore* IV GPU, dan 256 *Megabyte* RAM [7]. Penyimpanan data didesain tidak untuk menggunakan hard disk atau *solid state drive*, melainkan mengandalkan kartu SD (SD *memory card*) untuk *booting* dan penyimpanan jangka panjang.



Gambar 2. 1 *Raspberry PI* Model B beserta *Pin Out* [7]



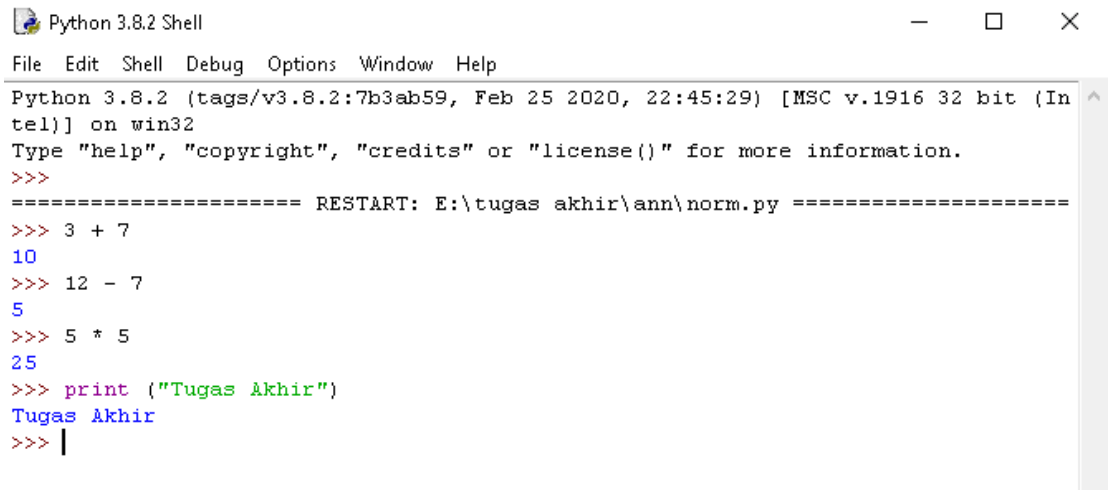
Gambar 2. 2 *Rasberry PI* Model A beserta *Pin Out*[7]

Hardware Raspberry Pi tidak memiliki *real-time clock*, sehingga OS harus memanfaatkan *timer* jaringan server sebagai pengganti. Namun komputer yang mudah dikembangkan ini dapat ditambahkan dengan fungsi *real-time* (seperti DS1307) dan banyak lainnya, melalui saluran GPIO (*General-purpose input/output*) via antarmuka I²C (*Inter-Integrated Circuit*).

Raspberry Pi bersifat *open source* (berbasis *Linux*), *Raspberry Pi* bisa dimodifikasi sesuai kebutuhan penggunanya. Sistem operasi utama *Raspberry Pi* menggunakan *Debian* GNU/*Linux* dan bahasa pemrograman *Python*. Salah satu pengembang OS untuk *Raspberry Pi* telah meluncurkan sistem operasi yang dinamai *Raspbian*, *Raspbian* diklaim mampu memaksimalkan perangkat *Raspberry Pi*. Sistem operasi tersebut dibuat berbasis *Debian* yang merupakan salah satu distribusi *Linux* OS [7]

2.2.3 Bahasa Pemrograman Python

Bahasa pemrograman *Python* ini pertama kali dibuat oleh Guido van Rossum pada awal tahun 1990 di negeri Belanda sebagai pengganti bahasa pemrograman yang disebut ABC. Walaupun Guido adalah orang yang pertama kali menciptakan bahasa pemrograman ini, tetapi bahasa pemrograman *Python* yang digunakan sekarang merupakan konstribusi dari berbagai sumber [8]. Bahasa pemrograman *Python* merupakan bahasa pemrograman yang dapat dikembangkan oleh siapa saja karena besifat *Open Source* atau dengan kata lain bahasa pemrograman ini gratis, dapat digunakan tanpa *lisensi*, dan dapat dikembangkan semampu yang dapat dilakukan. Sebenarnya bahasa pemrograman *Python* ini mudah dipelajari karena penulisan *sintaks* yang lebih *fleksibel*. Selain itu, bahasa pemrograman *Python* ini memiliki *efisiensi* tinggi untuk struktur data level tinggi, pemrograman berorientasi objek lebih sederhana tetapi efektif, dapat bekerja pada *multi platform*, dan dapat digabungkan dengan bahasa pemrograman lain untuk menghasilkan aplikasi yang diinginkan. *Python* dikenal sebagai bahasa pemograman *interpreter*, karena *Python* dieksekusi dengan sebuah *interpreter*. Terdapat dua cara untuk menggunakan *interpreter*, yaitu dengan mode baris perintah dan *modus script*. Pada mode baris, perintah diketikkan pada *shell* atau *command line* dan *Python* langsung menampilkan hasilnya. Bila menggunakan *shell*, semua definisi yang telah dibuat baik fungsi atau *variabel* akan dihapus. Cara lain adalah dengan menyimpan perintah – perintah *python* dalam satu *file*, yang disebut selanjutnya sebagai *script*. Kita dapat mengetikkan perintah-perintah *Python* dengan menggunakan *text editor* seperti *Notepad*. Lalu menyimpannya dengan akhiran ".py". kemudian menjalankannya dengan *Python* [9]. Pada gambar 2.3 ditunjukkan tampilan *Integrated DeveLopment Enviroment* (IDLE). *Prompt* >>>> menyatakan *interpreter Python* siap menerima perintah dari pemakai.



Gambar 2. 3 Tampilan IDLE *Python* pada *Windows* XP

Bahasa pemograman *Python* adalah bahasa pemograman yang mudah dibaca dan ter-struktur, hal ini karena di gunakannya sistem identasi. Yaitu memisahkan *blok* - *blok* program susunan identasi. Jadi untuk memasukan *sub* - *sub* program dalam suatu *blok*, *sub* - *sub* program tersebut diletakkan satu atau lebih *spasi* dari kolom suatu *blok* program.

2.2.4 Database Server

Database server adalah program komputer yang menyediakan layanan data lainnya ke komputer atau program komputer, seperti yang ditetapkan oleh model *client-server*. Istilah ini juga merujuk kepada sebuah komputer yang didedikasikan untuk menjalankan program server database. Database sistem manajemen database yang sering menyediakan fungsi server, dan beberapa DBMSs (misalnya, MySQL) secara *eksklusif* bergantung pada model *client-server* untuk akses data.

Client-server model dapat diartikan sebagai model dari suatu sistem yang membagi proses sistem antara server yang mengolah database dan *client* yang menjalankan *aplikasi*. Database server mengurangi beban akses data oleh *client* pada server. Database dapat diakses oleh beberapa *client* secara bersamaan dimana data yang diakses hanya atau diubah berasal dari satu sumber yaitu database pada server. Server tersebut diakses baik melalui suatu *front end* yang berjalan di komputer pengguna yang

menampilkan data yang diminta atau *back end* yang berjalan pada server dan menangani tugas-tugas seperti analisis data dan penyimpanan [10].
Dalam model *master-slave*, database server *master* adalah lokasi pusat dan utama data sementara database server pembantu disinkronisasi *backup* dari master bertindak sebagai *proxy* Beberapa contoh dari server basis data *Oracle*, DB2, *Informix*, *Ingres*, SQL Server. Setiap server menggunakan *query* sendiri *logika* dan struktur. Bahasa *query* SQL kurang lebih sama di semua server database.

2.2.5 Crontab

Cron adalah *daemon* penjadwalan yang menjalankan tugas dengan interval yang ditentukan. Tugas-tugas ini disebut *cron jobs* dan sebagian besar digunakan untuk mengotomatisasi pemeliharaan atau administrasi sistem. Kegunaan utama dari cron yaitu menjadwalkan *cron jobs* untuk dijalankan berdasarkan menit, jam, hari dalam bulan, tahun, hari dalam seminggu atau kombinasi dari semuanya.

Crontab (***cron table***) adalah file teks yang menentukan jadwal *cron jobs*. Ada dua jenis file crontab. File crontab seluruh sistem dan file crontab pengguna individu. File crontab pengguna disimpan dengan nama pengguna dan lokasinya bervariasi menurut sistem operasi. Dalam sistem berbasis *Red Hat* seperti *CentOS,* file crontab disimpan di direktori /*var/spool/cron* sementara pada file *Debian* dan *Ubuntu* disimpan di direktori *var/spool/cron/crontabs*. Mengedit crontab dengan menggunakan perintah crontab melalui baris perintah. /etc/crontab dan file di dalam direktori */etc/cron.d* adalah file crontab di seluruh sistem yang hanya dapat diedit oleh *administrator* sistem. Didalam *linux* peletakan *script* di dalam direktori */etc/cron*{*hourly*,*daily*,*weekly*,*monthly*} dan *script* akan dieksekusi setiap jam/harian/mingguan/bulanan. [11]

Lima bidang pertama dapat berisi satu atau lebih nilai, dipisahkan oleh koma atau rentang nilai yang dipisahkan oleh tanda hubung.

- * – Operator tanda bintang berarti nilai apa pun atau selalu. Jika memiliki simbol tanda bintang di bidang Jam, itu berarti tugas dilakukan setiap jam.

- , – Operator koma memungkinkan menentukan daftar nilai untuk pengulangan. Misalnya, jika memiliki 1,3,5 di bidang Jam, tugas akan berjalan pada jam 1 pagi, 3 pagi dan 5 pagi (format jam 24 jam).
- - Operator tanda hubung memungkinkan untuk menentukan rentang nilai. Jika memiliki 1-5 di bidang *Day of week,* tugas akan berjalan setiap hari kerja (Dari Senin hingga Jumat).
- / -Operasi garis miring memungkinkan Anda untuk menentukan nilai yang akan diulang selama *interval* tertentu di antara mereka. Misalnya, jika memiliki * / 4 di bidang Jam, maka tindakan akan dilakukan setiap empat jam. Ini sama dengan menentukan 0,4,8,12,16,20. Alih-alih tanda bintang sebelum operator garis miring, dapat menggunakan rentang nilai, 1-30/10 artinya sama dengan 1,11,21.