BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Pada penelitian Muhammad Sa'ad Rosyidi, 2017 dengan judul 6 "Rancang Bangun Alat Pembersih dan Penyortir Ukuran Telur Asin Berbasis Arduino Mega 25600'. Pada penelitian ini ketika sensor photodiode aktif kemudian konveyor menggerakkan telur asin yang selanjutnya dibersihkan menggunakan sikat yang diputar oleh motor dc dan disemprot dengan air menggunakan pompa dc. Proses penyortiran menggunakan load cell secara otomatis dapat menimbang berat telur selanjutnya dipindahkan dengan motor servo kemudian dihitung dengan sensor photodiode [6].

Menurut Armadani, Frengky Tarigan dkk Bebek merupakan salah satu ternak unggas yang berperan dalam menghasilkan telur dan daging. Selain murah bebek juga mudah ditemukan, bebek bukan hanya dimanfaatkan dagingnya saja melainkan telurnya juga bisa dimanfaatkan dalam berbagai macam hal seperti dikonsumsi, digunakan sebagai bahan tambahan untuk membuat kue, alat kecantikan dan bahan perekat. Masyarakat jarang mengkonsumsi telur bebek secara langsung dikarenakan baunya yang sangat amis dari pada telur unggas-unggas lainnya, kebanyakan masyarakat menjadikan telur bebek menjadi telur asin selain sedikit menghilangkan bau amis telur tersebut pengasinan telur juga bisa menjadikan telur bebek menjadi tahan lama. Bebek mempunyai kebiasaan yang buruk yaitu mudah gugup dan

bertelur disembarang tempat sehingga menyebabkan telur menjadi kotor karena lumpur dan kotoran yang menempel pada cangkang [7].

Pada penelitian Danang Subarkah Hadikawuryan, R.Imanu Danar Herunandi, dan Kriswanto, 2018 yang berjudul “Rancang Bangun Mesin Pencuci Telur Otomatis” bertujuan untuk menghemat energi UKM pengrajin telur asin maupun telur mentah. Penelitian ini dilakukan dengan objek berupa mesin pencuci telur ekonomis, dimana didapatkan hasil penelitian menggunakan motor listrik ¼ hp, kecepatan konveyor 0,225 m/mnt, daya pompa 100 watt terhubung dengan sikat pembersih pada putaran 2900rpm, konsumsi listrik Rp.237,-/jam, dengan kapasitas 307 butir/jam. Jumlah telur yang banyak tersebut sesuai dan cukup untuk kebutuhan konsumen [8].

Pada penelitian Wahyudi, Abdurrahman, dan Muhammad Nawawi, 2017 dengan judul ”Perbandingan Nilai Ukur Sensor Load Cell Pada Penyortir Buah” didapatkan hasil penelitian terhadap timbangan manual, kemudian dapat disimpulkan bahwa hasil pengukuran sensor load cell lebih efisien dan akurat dalam melakukan pengukuran berat buah. Hasil analisa data dari hasil kinerja keduanya diperoleh tingkat keberhasilan pengukuran load cell sebesar 97,73% sehingga tingkat kesalahannya hanya 2,27%, sedangkan tingkat keberhasilan pengukuran timbangan manual atau konvensional sebesar 97,34% sehingga tingkat kesalahannya lebih besar 2,64%. Metode membandingkan hasil ukur sensor load cell dengan timbangan manual merupakan salah satu cara yang efektif untuk mengetahui tingkat efisiensi dan akurasi sistem kerja dari keduanya [9].

Pada penelitian Irfan Fauzi Aristianto, Mohamad Ramdhani, IG.Prasetya Dwi Wibawa, 2020 yang berjudul “Rancang Bangun Sistem Sortir Telur Ayam” diperoleh hasil bahwa penggunaan sensor load cell dan motor servo cocok pada sistem ini untuk meminimalkan biaya produksi alat. Sensor load cell 1 kg didapat nilai akurasi sensor sebesar 99,25% saat koveyor dalam kondisi diam, dan meningkat menjadi 99,28% saat konveyor dalam kondisi berjalan. Berdasarkan hasil akurasi yang tinggi tersebut maka penyortiran ayam berhasil. Selain itu, penggunaan motor servo sebagai penyearah telur sesuai dengan kelompoknya juga memiliki tingkat keberhasilan yang tinggi sebesar 100%. Penggunaan mekanik conveyor juga memudahkan peternak sehingga tidak perlu memilah secara manual dan menghasilkan nilai akurasi yang lebih tinggi karena dalam 1 menit dapat menyortir 14 telur tanpa retak sesuai dengan kelompoknya dengan kecepatan motor 72.5 rpm serta delay sensor load cell saat menimbang selama 1 detik. Daya yang diperlukan sebesar 191,2 watt sehingga cocok untuk peternak ayam petelur rumahan yang tidak memerlukan daya besar agar alat berjalan [10].

2.2 Landasan Teori

2.2.1 Sistem Monitoring

Perancangan Sistem monitoring atau sistem pengawasan adalah suatu upaya yang sistematik untuk menetapkan kinerja standar pada perencanaan untuk merancang sistem umpan balik informasi, untuk

membandingkan kinerja aktual dengan standar yang telah ditentukan, untuk menetapkan apakah telah terjadi suatu penyimpangan tersebut, serta untuk mengambil tindakan perbaikan yang diperlukan untuk menjamin bahwa semua sumber daya perusahaan atau organisasi telah digunakan seefektif dan seefisien mungkin guna mencapai tujuan perusahaan atau organisasi [11].

2.2.2 Mit App Inventor

Merupakan platform untuk memudahkan proses pembuatan aplikasi sederhana tanpa harus mempelajari atau menggunakan bahasa pemrograman yang terlalu banyak. Kita dapat mendesain aplikasi android sesuai keinginan dengan menggunakan berbagai macam layout dan komponen yang tersedia [12].

Gambar 2.1 Mit App Inventor

2.2.3 Pengertian *Android*

Android berbasis pada kernel *Linux* dan menggunakan bahasa pemrograman *Java* sebagai bahasa utama untuk mengembangkan

aplikasi. *Android* menyediakan *API* (*Application Programming Interface*) yang memungkinkan para pengembang untuk membuat aplikasi yang dapat diinstal dan dijalankan pada berbagai perangkat *Android*.

Salah satu keunggulan *Android* adalah kemampuan untuk disesuaikan dengan berbagai jenis perangkat. *Android* mendukung berbagai tipe layar, seperti layar sentuh (*touch screen*), resolusi layar yang berbeda, dan dukungan terhadap berbagai sensor seperti *GPS*, *akselerometer*, dan kamera. Selain itu, *Android* juga memiliki banyak fitur seperti notifikasi, manajemen file, *multitasking*, dan pengaturan privasi [13].

Android juga memiliki *platform Google Play Store* yang meyajikan berbagai aplikasi maupun *game* yang dapat diunduh oleh pengguna *smartphone Android*. Para pengembang juga dapat mengembangkan aplikasi *Android* dan memasarkannya di *Google Play Store*, sehingga memungkinkan mereka untuk mencapai pasar yang lebih luas.

Gambar 2.2 Logo Android

2.2.4 Pengertian *Database MySQL*

Database adalah suatu kumpulan data yang disusun sedemikian rupa sehingga membentuk informasi yang sangat berguna. *Database* berbentuk dari sekelompok data yang memiliki jenis/sifat yang sama. Demikian juga, kumpulan dari data-data penjualan, keuangan dan lainnya dapat dikumpulkan lagi menjadi kelompok besar, misalnya data-data perusahaan. Dalam berkembangnya data tersebut dapat berbentuk sebagai macam data, misalnya dapat berupa program, lembaran *entry* (memasukan) data, kesemuanya itu dapat dikumpulkan menjadi satu yang disebut dengan database. Salah satu *database* yang populer adalah *SQL* [14].

Gambar 2.3 Tampilan Logo MySQL

2.2.5 Pengertian *XAMPP*

Xampp adalah distribusi apache kecil dan ringan yang mengandung teknologi pengembangan *web* yang paling umum dalam satu paket. sedangkan *Xampp* menurut beberapa ahli merupakan paket progam web lengkap yang dapat dipakai untuk belajar pemprograman *web*, khususnya *php* dan *mysql* paket ini dapat didownload secara

gratis dan legal. *Web server* ini yang mudah digunakan serta dapat melayani tampilan halaman *web* yang dinamis dan dapat diakses secara local menggunakan *web server* local (*localhost*), setelah melakukan install *software* pendukung *web server* yaitu *Apache*, *PHPMyAdmin*, *database MySQL* dan *web server* ini bersifat instan (siap digunakan) yang dapat berjalan pada sistem operasi Linux dan sistem operasi Windows [15].

2.2.6 UML (*Unifield Modeling Language*)

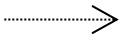
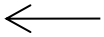
Unifield Modeling Language merupakan salah satu metode pemodelan visual yang digunakan dalam perancangan dan pembuatan sebuah *software* yang berorientasikan pada objek. *UML* merupakan sebuah standar penulisan atau semacam *blue print* diamna didalamnya termasuk sebuah bisnis proses, penulisan kelas-kelas dalam sebuah bahasa yang spesifik. Terdapat beberapa diagram *UML* yang sering digunakan dalam pengembangan sebuah sistem. Terdapat beberapa diagram *UML* yang sering digunakan dalam pengembangan sebuah sistem yaitu:

1. *Use Case* Diagram

   Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* mempresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke

sistem, meng-create sebuah daftar belanja, dan sebagainya. Seorang atau sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan - pekerjaan tertentu.

Tabel 2.1 Simbol Use Case Diagram

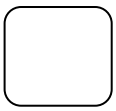
No	Gambar	Nama	Keterangan
1.		Actor	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan use case.
2.		Dependency	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (independent) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (independent).
3.		Generalization	Hubungan dimana objek anak (descendent) berbagi perilaku dan struktur data dari objek yang ada diatasnya objek induk (ancertor).
4.		Include	Menspesifikasikan bahwa use case sumber secara eksplisit.
5.		Extend	Menspesifikasikan bahwa use case target memperluas perilaku dari use case sumber pada suatu titik yang diberikan.
6.		Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7.		System	Menspesifikasikan paket yang menampilkan sistem secara terbatas.

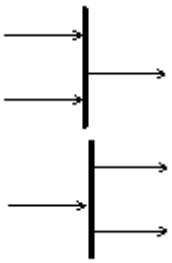
No	Gambar	Nama	Keterangan
8.		Use Case	Deskripsi dari urutan aksi – aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
9.		Collaboration	Interaksi aturan – aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen – elemennya (sinergi).
10.		Note	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

2. *Activity* Diagram

Sebuah diagram aktivitas UML menggambarkan perilaku dinamis dari suatu sistem atau bagian dari sistem melalui aliran kontrol antara aksi yang dilakukan sistem.

Tabel 2.2 Simbol Activity Diagram

No	Gambar	Nama	Keterangan
1.		Activity	Memperlihatkan bagaimana masing - masing kelas antarmuka saling berinteraksi satu sama lain.
2.		Action	State dari sistem yang mencerminkan eksekusi suatu aksi.
3.		Initial Node	Bagaimana objek dibentuk atau diawali.
4.		Final Node	Pilihan untuk mengambil keputusan

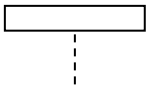
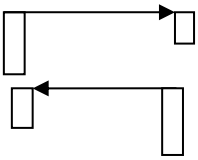
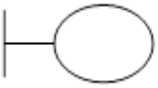
No	Gambar	Nama	Keterangan
5.		Fork Node	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran.
6.		Decision	Pilihan untuk mengambil keputusan
7.		Fork/Join	Digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu
8.		Rake	Menunjukkan adanya dekomposisi
9.		Time	Tanda waktu
10.		Send	Tanda pengiriman

3. Sequence Diagram

Sequence diagram menggambarkan interaksi antar di sekitar (pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. *Sequence* diagram terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek - objek yang terkait). Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah - langkah yang dilakukan sebagai respon dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara

internal dan output apa yang dihasilkan. Masing - masing objek, termasuk aktor, memiliki lifeline vertical.

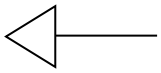
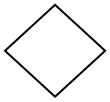
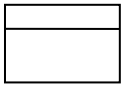
Tabel 2.3 Simbol Sequence Diagram

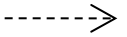
No	Gambar	Nama	Keterangan
1.		LifeLine	Objek entity, antar muka yang saling berinteraksi.
2.		Message	Spesifikasi dari komunikasi antar objek yang memuat informasi – informasi tentang aktifitas yang terjadi.
3.		Actor	Menggambarkan orang yang sedang berinteraksi dengan sistem
4.		Boundary Class	Menggambarkan penggambaran dari form
5.		Entity Class	Mengambarkan hubungan kegiatan yang akan dilakukan
6.		Control Class	Menggambarkan penghubung antara Boundary dengan tabel
7.		Activation	Sebagai sebuah objek yang akan melakukan sebuah aksi
8.		Message	Mengindikasikan komunikasi antara objek dengan objek
9.		Self Message	Menginndikasikan komunikasi kembali kedalam sebuah objek itu sendiri

4. Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. Class memiliki tiga area pokok : nama (*stereotipe*), atribut, dan metoda.

Tabel 2.4 Simbol Class Diagram

No	Gambar	Nama	Keterangan
1.		Generalization	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*)
2.		Nary Association	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3.		Class	Himpunan dari objek - objek yang berbagi atribut serta operasi yang sama.
4.		Collaboration	Deskripsi dari urutan aksi - aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
5.		Dependency	Operasi yang benar - benar dilakukan oleh suatu objek.

No	Gambar	Nama	Keterangan
6.		Dependency	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
7.		Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.