IR MAN
Design Document
for ECE Capstone Project

Max Bai, Shirley Zhang, Jiaqi Zou
Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—IR MAN, a universal IR Controller on a raspberry pi connected to remote web server, and also as an interactive humanoid robot with 2 degrees of freedom (DOF) motion, computer vision (CV) based object detection and IR control capabilities. The Internet of things (IoT) element of this project is driven by the convergence of multiple technologies such as machine learning, cloud services, commodity sensors, and real time embedded systems, hence achieving the goal of remotely controlling your household appliances via the internet. The following sections of this paper elicit the motivation, architecture design, technical requirements and testing metrics of this particularly innovative approach to deploying this IoT Smart Home System (IoTSHS). Using computer vision based approach, the system (IR MAN) can effectively identify the exact location of the specified household appliances and plan motion accordingly to point the IR emitter directly at it before emitting its corresponding IR commands. The proposed IoTSHS will be designed, programmed, manufactured, integrated and tested in the format of specifications in this paper.**

***Index Terms*—IR Control, Computer Vision, Internet of Things, Robot Dynamics, AWS Deployment, Home Appliances, Smart Home**

I. Introduction

THE "Internet of things" (IoT) is becoming an increasingly growing topic of conversation in the entire tech industry. With Broadband Internet becoming more and more accessible, the cost of connectivity is dramatically decreasing and mobile devices like smartphones has a sky-rocketed its penetration to the market. All of these things are creating a "perfect storm" for the IoT. However, many of our current household Infrared (IR) controlled appliances are "traditionally dumb" devices that are not internet capable. Thus, the problem arises when we aspire to come up with a scalable approach to empower these devices with IoT capabilities under affordable cost and extensive add-on values.

Have you ever ran into the problem of not being able to find that remote control lying somewhere under the couch to turn on your TV? Or too many remote controls and you dont know which one correspond to which device? Have you imagined being able to turn on your AC just before you got home on a hot summer day? With so many domestic appliances controlled by IR remote control, we decided to create IR MAN to make our life easier by designing a smart IoT IR Control Hub that allows your smartphone to remotely control all the traditionally dumb or non-internet-enabled IR domestic appliances. The solution proposed in this paper, IR MAN, aims to solve the aforementioned problems such that the remote control is truly remote and all home appliances are automatically upgraded to IoT devices with minimal cost. And more importantly, IR MAN is an interactive humanoid robot that actively locates the household appliances in your house using computer vision (CV) and plans motions accordingly to reorient the IR emitter in order to accurately shoot the IR command signals at it like a real Iron Man.

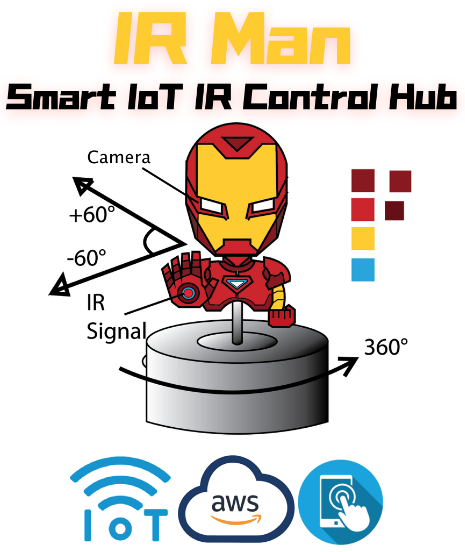


Fig. 1: Physical Design for IR MAN

II. Design Requirements

A. *Software Requirements*

The main requirement of our product is latency and user experience: users need to be able to control IR devices successfully and quickly. With regard to this principle, we specified requirements for each components separately.

For the webapp server, the UI interface should be responsive. Moreover, the webapp to server connection latency should be less than 500ms, which ensures that there is little to no delay between the time when user pressing a button on the webapp and the time when IR signal reaches the device.

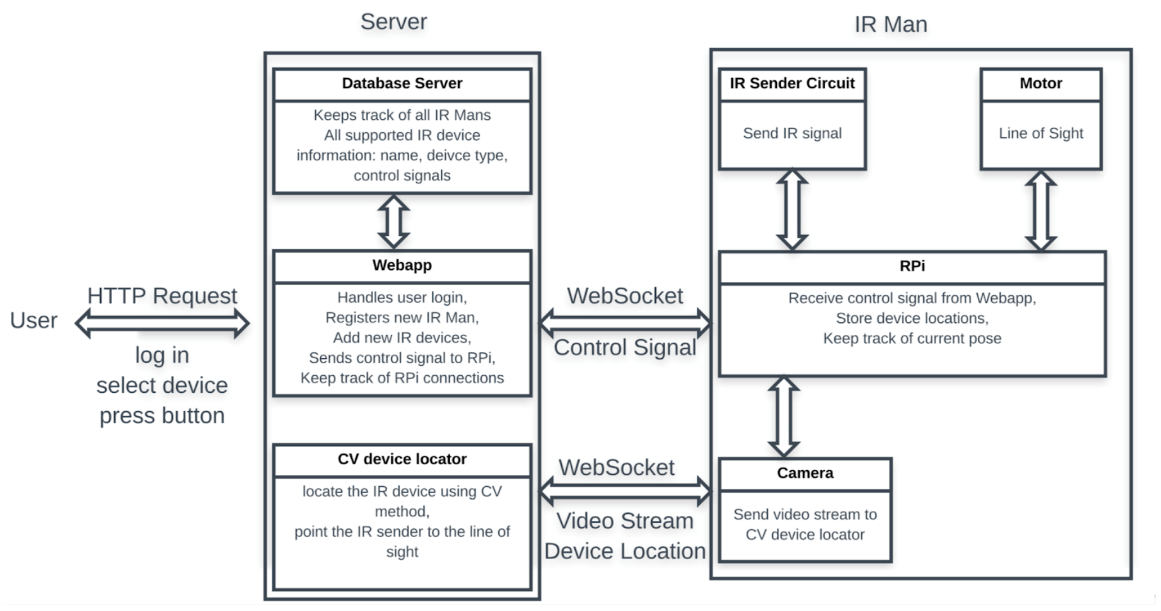

Server
IR Man
Database Server
Keeps track of all IR Mans
All supported IR device information: name, deivce type, control signals
Webapp
Handles user login,
Registers new IR Man,
Add new IR devices,
Sends control signal to RPi,
Keep track of RPi connections
CV device locator
locate the IR device using CV method,
point the IR sender to the line of sight
IR Sender Circuit
Send IR signal
Motor
Line of Sight
RPi
Receive control signal from Webapp,
Store device locations,
Keep track of current pose
Camera
Send video stream to CV device locator
User
HTTP Request
log in
select device
press button
WebSocket
Control Signal
WebSocket
Video Stream
Device Location

Fig. 2: System Architecture for IR MAN

In terms of the device locator server, we expect the major latency to be sending the image over websocket connection and actually running the object detection model. For the image transmission, we are aiming at a latency less than 500ms. For the overall device locating pipeline, we are aiming for less than 3 minutes, knowing that we plan to take 72 images for one round and the processing throughput of project detection model on CPU is around 1 image per second after heated to appropriate temperature, and that the first few images is going to take way longer. This computer vision process only needs to be done when the device is being set up. Since user only has to do this only once, we think that the 3 minute requirement does NOT affect user experience. The accuracy of the resulting estimated device location needs to be within 10 degrees of the actual device location. Moreover, for the object detection algorithm, we intend to achieve a validation accuracy of more than 75%.

B. Hardware Requirements

There are mainly two hardware components in our project: the IR circuit and the motors. For the IR circuit, we are aiming for greater than 90% success rate of controlling IR devices using our IR circuit under different circumstance, such as lighting and relative location of the IR device to the IR circuit.

For the motors, given a specific pose, the motors needs to be within 5 degrees of the correct pose. This is because we are taking 72 images for 360 degree, which means each image covers 5 degree. And for the sake of latency, time it takes from a random pose to reach the desired location should be under 2s.

C. Overall Requirements

In terms of the user experience of our whole system as a final product, we want to be able to achieve the following requirements. Time that it takes for any IR signal command needs to be sent in less than 3 seconds. And the average success rate of controlling all IR device needs to over 90%. The time to run the complete computer vision device locating service should be under 3 minutes. And the accuracy of each device location needs to be within 10 degrees of the true location.

The general design of the whole system is illustrated in the system architecture diagram in Fig 2. From the user end to the device end, we have the following major components: the webapp server which handles all the user interactions, the device locator server which handles computer vision inferences, and a Raspberry Pi that controls all the peripherals and websocket connections. Both of the two servers are running on the cloud. User can log into the webapp, find a list of all IR devices, and go to the control panel of a specific device and control the device just like a remote controller. The device locator server receives video stream from RPi and runs object detection algorithm as a backstage process. The Raspberry Pi will be interfacing with the camera modules, IR circuit and motors and motor controllers.

On the IR Man device, an RPi is used as the control hub for motor, camera and the IR Circuit. Upon first-time calibration, RPi sends frames from camera module to device locator server as the robot spins. The device locator gets back the corresponding device locations and RPi stores these locations locally. Upon a new button press, the RPi starts the motor to point to the specific device and then send the IR signal through the IR Circuit. The motor controls the pose of the robot and the IR Circuit sends any IR signal from RPi. There are several communication pathways in the system:

1) From webapp server to RPi: The webapp server to RPi communication handles the following kinds of requests: (1) user presses a button on Webapp to control a specific device, (2) user orders IR Man to find device locations. The communication protocol includes an opcode to identify the request type and specific content of the request. For the first kind of request where a user presses a button, the message includes the brand and type of the device, and the button name and the control signal hex code. For the second type of request, the messages includes the list of devices to search for in the room.

2) From RPi to device locator: When user instruct the IR Man to search for device locations, RPi sends the video stream to device locator on AWS as following bundle: $(frame, \theta_1, \theta_2)$. This bundle identifies the objects that the RPi is seeing at different pose. The frame is sent using ImageZMQ, an image serialization protocol. We adopted ImageZMQ because it is a high-performance asynchronous message passing library for many distributed systems. We chose to use ImageZMQ because it has the least memory copy, which helps with latency. With this library in place, our RPi will be able to transfer the images and stream videos with OpenCV over the network with a smaller latency and higher throughput.

3) Between RPi and camera: The camera module we adopted is Raspberry Pi's official product, the Camera Module v2. The connection to RPi is very straight forward, which is to connect the 15 pin FPC directly with the dedicated CAM socket on Raspberry Pi. It attaches to Pi by way of one of the small sockets on the board upper surface and uses the dedicated CSI interface, designed especially for interfacing to cameras. With 8 megapixels of still resolution and 1080p30, 720p60 and 480p90 for videos. Its horizontal field of view of 62.2 degrees and vertical field of view of 48.8 degrees will allow a better accuracy for our device locator algorithm since the accuracy was previously limited by the number of frames provided, dictated by the network bandwidth. However, with wider field of view, the possibility of target objects appearing in each frame is larger, thus reducing the amount of frames to be transmitted over the network.

4) Between RPi and IR circuit: The IR circuit is composed of IR LEDs and MOSFETs. The MOSFET is a voltage controlled current sources that dictates the I_D that flows through it. The ON and OFF of the MOSFET is controlled by one of the programmed GPIO pins on Raspberry Pi. The specific ON/OFF PWM signal is generated using an open source IR signal database called LIRC that collected almost all brand's IR signal protocols. The protocols are enclosed in some hex codes representing bitmaps of ON/OFF levels in a given duty cycle. This database works with LIRC [1] (Linux Infrared Remote Control) package on RPi, which is capable of wrapping up the above process with command line arguments. We will be using an Open source IR remote control library: LIRC to controls the IR circuit. The IR Signals can also be retrieved from this library.

5) Between RPi and motor: In order to support the required motion of IR MAN, we need our RPi to control both a stepper motor and a servo. After the aforementioned computer vision based device locator algorithm returns the device location in terms of (θ_1, θ_2). θ_1 represents the target position of the motor in the chassis base. θ_2 represents the target position of the motor in the arm of IR MAN. Our RPi will then run a motion planning script we wrote to drive the motor/servo to the target position specified by (θ_1, θ_2). To control the stepper motor from RPi, we acquired a separate controller board called "Easy Driver" from SparkFun. The driver is essentially an H-bridge that controls the spin direction and duty cycles for the stepper motor's voltage level.

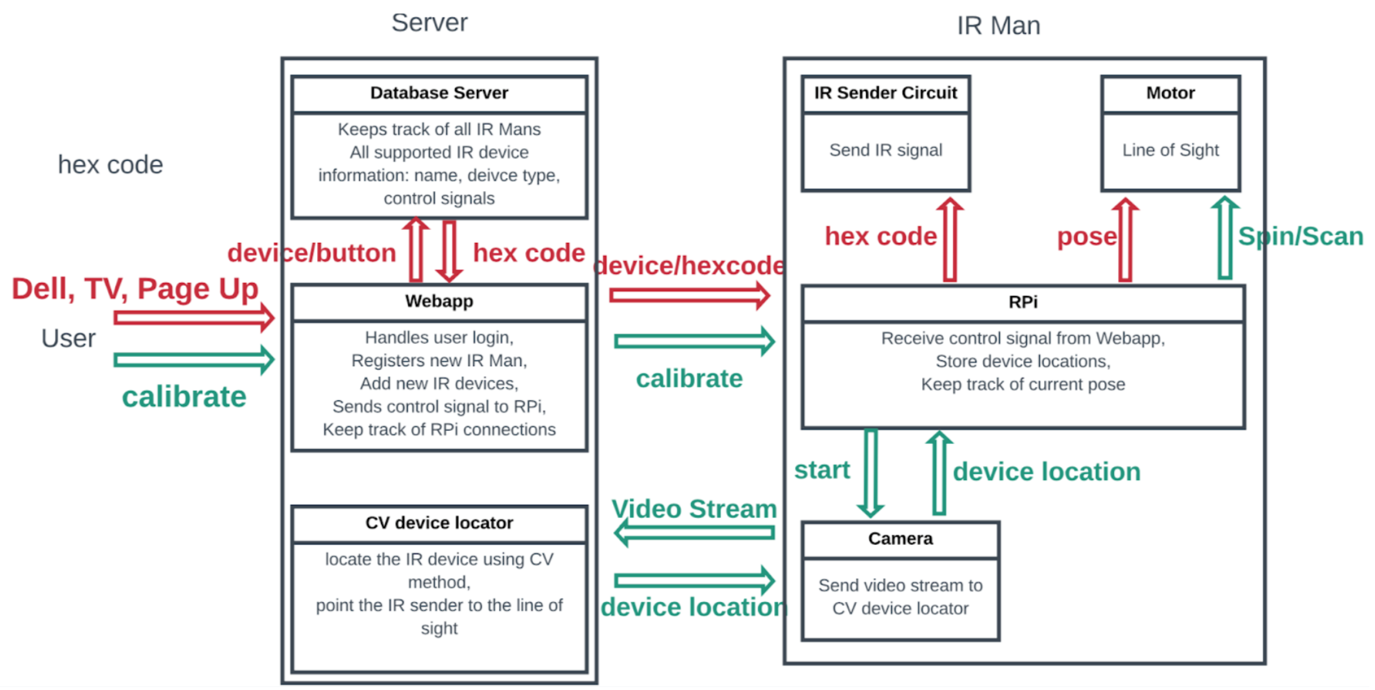

Server
IR Man
hex code
Database Server
Keeps track of all IR Mans
All supported IR device information: name, deivce type, control signals
device/button
hex code
Dell, TV, Page Up
User
calibrate
Webapp
Handles user login, Registers new IR Man, Add new IR devices, Sends control signal to RPi, Keep track of RPi connections
device/hexcode
calibrate
CV device locator
locate the IR device using CV method, point the IR sender to the line of sight
Video Stream
device location
IR Sender Circuit
Send IR signal
Motor
Line of Sight
hex code
pose
Spin/Scan
RPi
Receive control signal from Webapp, Store device locations, Keep track of current pose
start
device location
Camera
Send video stream to CV device locator

Fig. 3: System Interaction for IR Man

D. *System Interaction*

To understand how the system components interact with each other, we introduce two major system interaction in our system: (1) user presses a button on the control panel in the webapp to control the IR device. (2) user orders the IR Man device to find the location of all IR devices. These two interactions are illustrated in System Interaction Diagram in Fig 3 using two different colors of arrow: red arrows for button press, green arrows for locating devices.

1) Button Press: The user send the button press HTTP request to the webapp server, upon receiving the request, the webapp server fetch the button information from the database server, which includes the device name, the device type, the protocol name, the button name, the button hexcode. The webapp server then find the user's IR Man device websocket connection by the IR Man device's hashcode. The button information is sent to the RPi over the websocket connection. Upon receiving the button information from webapp server, RPi extract the device name and find the predetermined device location, then it controls the motor module to move to correct pose for the device, so that the IR sender points to the device. Finally, RPi sends the control signal to the IR circuit.

2) Locating device: Before user can use the IR Man to control any IR devices, the IR Man needs to find the device locations in the terms of (θ_1, θ_2) as the pose for the tow motors. This bootstrap step also needs to be done whenever the user moves the IR Man to a new position. When user presses the "look for device" button on the webapp, the webapp server sends the list of user's IR device to RPi to search for these devices in the surroundings. The Rpi starts to spin the motor and send the video stream to the device locator server. The device locator server sends back specific location for each device to RPi. These locations are stored in non-volatile memory so that Rpi "remembers" the pose for these devices for future uses.

III. System Description

A. *Webapp Design*

For the webapp, we need to handler user authentication, adding new devices, choosing devices and button press. The following is the user interaction diagram and routing table for the webapp.

URL	Path	Used for
GET/POST	/login	User login
GET	/logout	User logout
GET/POST	/register	Register new user
GET	/devices	Display user's device list
GET/POST	/devices/new	Add new devices
GET	/devices/:id	Go to device control panel
POST	/:deviceID/:buttonName	Sending control signal to RPi
POST	/calibrate	Start the computer vision calibration

Fig. 4: Routing Table for the webapp

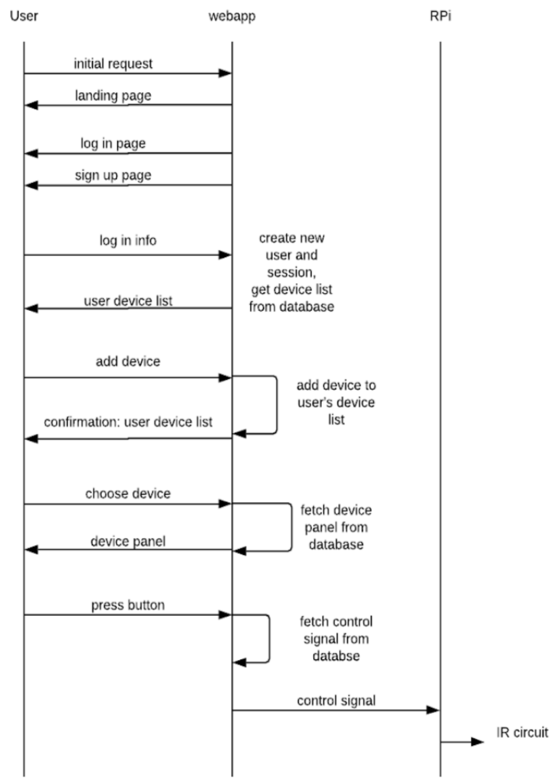

User
webapp
RPi
initial request
landing page
log in page
sign up page
log in info
create new user and session, get device list from database
user device list
add device
add device to user's device list
confirmation: user device list
choose device
fetch device panel from database
device panel
press button
fetch control signal from databse
control signal
IR circuit

Fig. 5: User Interaction Diagram for the webapp

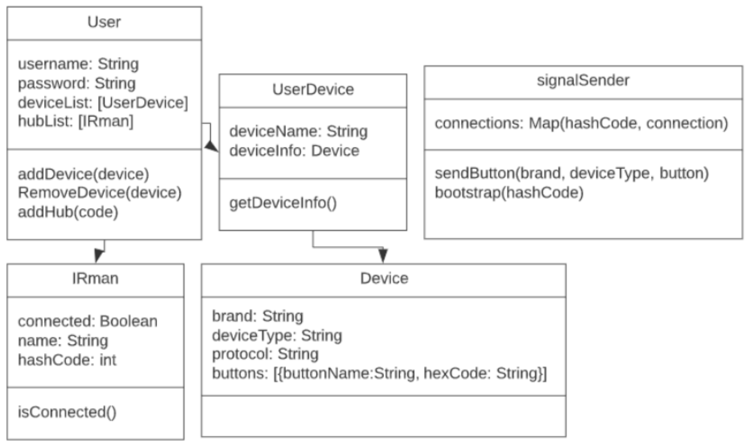
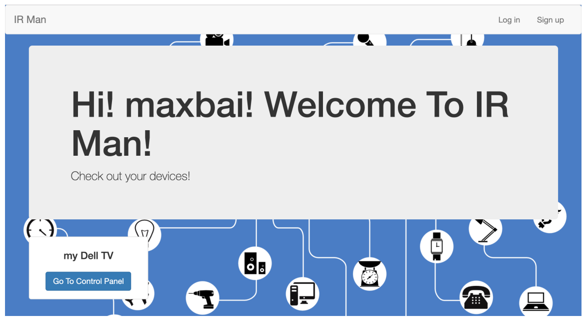


Fig. 6: Webapp Object Diagram for the webapp



Hi! maxbai! Welcome To IR Man!

Check out your devices!

Fig. 7: User Interface for the webapp

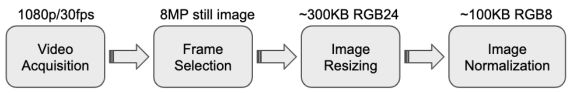


Fig. 8: Image Pre-processing Pipeline

For the object diagrams we have the following objects: User, UserDevice, Device, IRman and SignalSender. All these objects are stored in the database.The User object keeps track of the UserDevice and the IR Man device each user has. The Device keep tracks of the device information: what buttons does this device have and the their specific signals. The UserDevice has reference to its Device. For example, 5 users might be using the same Dell TV, and there is only one dell TV Device object to keep track of all the Dell TV's buttons. There are 5 different UserDevice that represents each user's dell TV. The signal sender keeps track of all connections to all RPi's and handles sending signal to RPi. Upon button press, the webapp would delegate the signal sending procedure to this object.

Our front-end user interface looks like the Fig 7, which is the displaying the all the devices for an user. The authenticated user can also control registered devices on this portal.

B. RPi Hardware Peripherals And Schematics

Our RPi controls all the additional hardware peripherals, namely the IR circuit, EasyDriver connected to a Nema Stepper Motor, a servo and a camera. Refer to Fig 9 on the next page. Details description of the schematics can be found in $Section\ V.B$ and $V.D$ Implementation Plan and tools.

C. Computer Vision Pipeline

The computer vision(CV) pipeline mainly consists of 3 stages: pre-processing stage, image inference stage and post-processing stage. Pre-processing prepares the gathered raw frames such that they may be fed into neural network in the right form. During the inference stage, the processed frames are fed into neural network, and produce outputs containing information of device location and device type on frame. After all frames are inferred, the gathered data then run through the post-processing stage, outputting device locations in the form of motor angles.

1) Image Pre-processing: The pre-process stage consists of image resizing and normalization, and are to be performed respectively on RPi and remote server. Reason for such arrangement and resulting network protocols(ImageZMQ) have been discussed above, and therefore will not be explained here. The resized images will be in the ratio of 16:9 (in proportional to the original 1080p frames), and after normalization, pixels will be in the form of intensity values ranging from 0 to 1. Therefore, each image will be in the form of a matrix of size $3 \times$height$\times$width, where 3 stands for 3 colour channels, containing floats in the range of $[0, 1]$. Refer to Figure 8 for the image pre-processing pipeline.

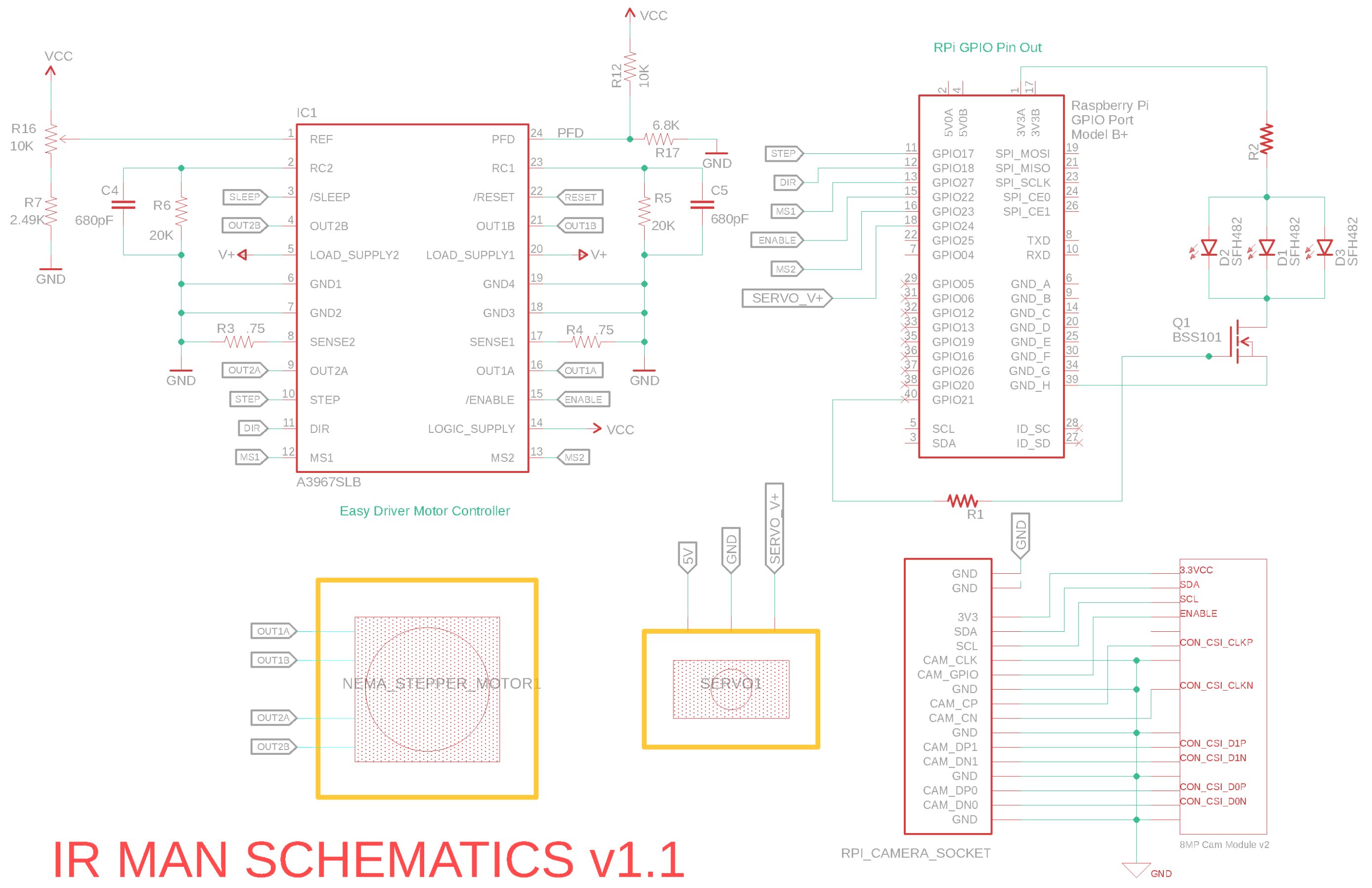


Fig. 9: Schematics for IR Man

2) Object Detection Algorithm: Just like the well known metaphor suggests, it is best to view the neural network framework as a bootstrapped blackbox. It is not necessary to understand what each layer is doing in the algorithm for the sake of our project, so we'll skip the hideous part and introduce the output instead. In OpenCV, The output of one image through object detection model will consist of 3 types of information: tag of the detected object, confidence of this result, and the bounding boxes of the detected object (in terms of upper left and lower right anchor coordinates). In case of multiple devices in one image, by default the output will be a list at the size of 100, each containing a piece of information of a possibly detected device, or -1 if no more device is detected. This size of output is a set value in OpenCV, and not to be changed.

3) Data Post-processing & Location Parsing: This step is to be performed only after all frames had been fed into the neural network model and output data collected. Since each frame has its corresponding motor positions(θ) transferred together, we can find the median value of those motor positions where a specific device (outputs with the same tag) shows up, and can take it as the correct orientation for this device. Note that we will only take outputs with confidence bigger than 0.5 into calculation. This threshold is often chosen as a standard approach in industry. Refer to Figure 10 for the image post-processing pipeline.

IV. Implementation Plan and Tools

A. WebApp Server

For the webapp server, we are using Express.js and Node.js as the web framework and MongoDB as the database server. For front-end, since we don't have dynamic content, we are simply using HTML/CSS and ejs. For user authentication, we are using Passport.js to handle user registration and logging in/out. The webapp server is going to be deployed to AWS EC2. We are writing all the routing logic, front-end interfaces, back-end logic, the communication protocl to RPi by ourselves. We plan to first implement the MVP webapp server on localhost before deploying it to EC2, since we only 750 hours per month on EC2, and we want to save that time for later system integration/testing and project demo.

B. IR circuit

The IR circuit is composed of an IR LED and a MOSFET that dictates the I_D that flows through it. The ON and OFF of the MOSFET is controlled by one of the programmed GPIO pins on Raspberry Pi. When that GPIO pin outputs a $3.3V$, that voltage level will be immediately fed into V_{GS}, which turns on the MOSFET, hence allowing current to flow through the IR LED, vice versa. The specific ON/OFF PWM signal is generated using an open source IR signal database called LIRC that collected almost all brand's IR signal protocols.

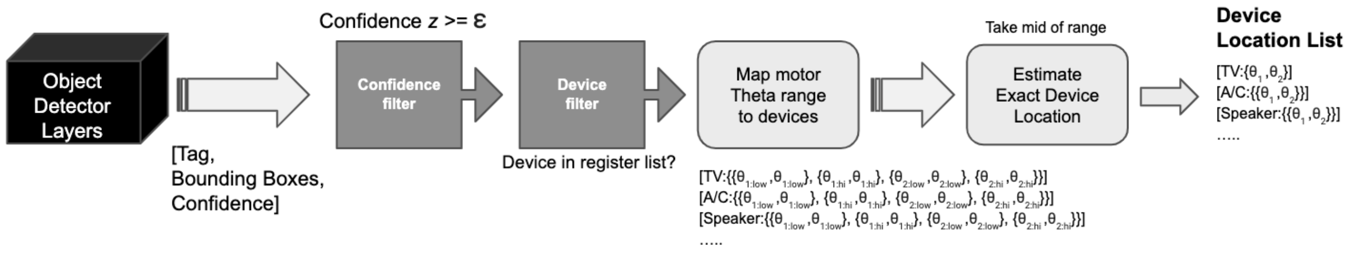

Object Detector Layers
[Tag, Bounding Boxes, Confidence]
Confidence $z >= \varepsilon$
Confidence filter
Device filter
Device in register list?
Map motor Theta range to devices
[TV:{{$\theta_{1:low}$,$\theta_{1:low}$}, {$\theta_{1:hi}$,$\theta_{1:hi}$}, {$\theta_{2:low}$,$\theta_{2:low}$}, {$\theta_{2:hi}$,$\theta_{2:hi}$}}]
[A/C:{{$\theta_{1:low}$,$\theta_{1:low}$}, {$\theta_{1:hi}$,$\theta_{1:hi}$}, {$\theta_{2:low}$,$\theta_{2:low}$}, {$\theta_{2:hi}$,$\theta_{2:hi}$}}]
[Speaker:{{$\theta_{1:low}$,$\theta_{1:low}$}, {$\theta_{1:hi}$,$\theta_{1:hi}$}, {$\theta_{2:low}$,$\theta_{2:low}$}, {$\theta_{2:hi}$,$\theta_{2:hi}$}}]
.....
Take mid of range
Estimate Exact Device Location
Device Location List
[TV:{θ_1,θ_2}]
[A/C:{{θ_1,θ_2}}]
[Speaker:{{θ_1,θ_2}}]
.....

Fig. 10: Image Post-processing Pipeline

The protocols are enclosed in some hex codes representing bitmaps of ON/OFF levels in a given duty cycle. This database works with LIRC (Linux Infrared Remote Control) package on RPi, which is capable of wrapping up the above process with command line arguments.The IR circuit is going to be designed, built, soldered and tested by ourselves. We will acquire the IR LED Diodes, MOSFETs, resistors and other basic electronic components from CMU ECE's circuit labs.We will be using an Open source IR remote control library: LIRC to controls the IR circuit. The IR Signals can also be retrieved from this library.

C. Computer Vision

To make sure that IRMAN recognizes the specific devices we provide, we will need to train our own object detection network with customized dataset and tags. In this case, we will not be constructing the neural network framework from scratch, but use Darknet, a pretrained neural network framework, and YOLOv3 configurations to further train it. The training process has been wrapped up cleanly and could be done through simple command line arguments under python openCV environment. What we need to provide is the dataset of images containing devices, with their location and category annotated in required format. This dataset will be collected through taking photos and acquiring existing photos from Google's OpenImagesV4 dataset.

D. RPi motor control

The motion of the IR Man is controlled by a stepper motor and a servo. The stepper motor is controlled by a motor controller board called EasyDriver acquired from SparkFun. The EasyDriver is compatible with anything that can output a digital 0 to 5V pulse. The EasyDriver requires a 9V to 30V supply to power the motor and can power any voltage of stepper motor. The EasyDriver has an on board voltage regulator for the digital interface that can be set to 5V or 3.3V, which would work perfectly on our Raspberry Pi. Since Nema Stepper Motor 17 is a bi-polar motor, it has 4 wires coming out that goes into the EasyDriver. Each pair corresponds to controlling the inductor coil. On the other hand, EasyDriver has 5 pins that connects to the RPi, which are STEP, DIR, MS1, MS2 and ENABLE. In particular, MS1 and MS2 pins broken out to change micro-stepping resolution to full, half, quarter and eighth steps (defaults to eighth), which can provide us precision turning. For the servo connection is a lot more straightforward. It connects a 5V, ENABLE and GND to the RPi.

E. Mechanical Design

For the mechanical design, we have thought of the following ways around the problem with wiring around a rotatable base. The IR circuit and camera are mounted on the body of the IR Man through wires. The camera, in particular, is connected through FPC cable, which is not twistable at all. The first step into solving this is to have our RPi sit on the same platform as IR Man figurine so that the IR Circuit and the Camera should stay in place with respect to relative position. Now the problem is how to go about the wiring interface between our RPi and the Nema stepper Motor that is not rotating with the RPi. We came up with the following approaches.

1. Have the Nema Motor sit upside down. In this way, the base of the motor is attached to the rotatable chassis and wires between RPi and Motor would no longer be a problem. (This would require Software changes and extra control algorithms to ensure accuracy)

2. Using a gear train to make the shaft spin off-centered from the round chasis and use a slip ring in the center of the chassis to solve cable entanglement issues.

3.Use an extra nRF24L01+ 2.4GHz transceiver module to enable wireless communication between an RPi and Arduino. While the RPi is spinning on the chassis with IR Man, it sends the command to an Arduino that drives the Nema Stepper Motor and not spinning. Since cables are eliminated, the problem is solved. (Extra Hardware Components. Complicated Wireless Communication with added latency. A separated software stack to be added that adds significant complexity to the system.)

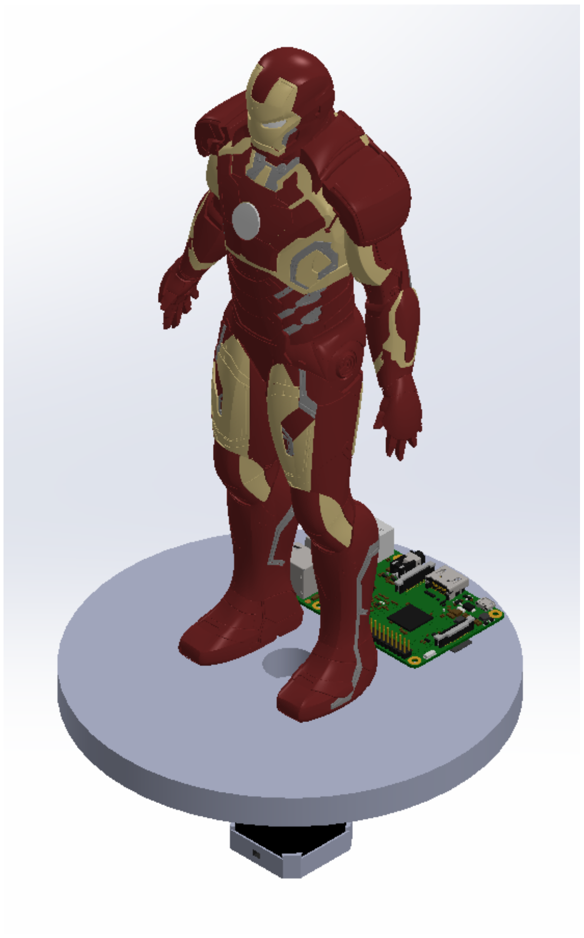
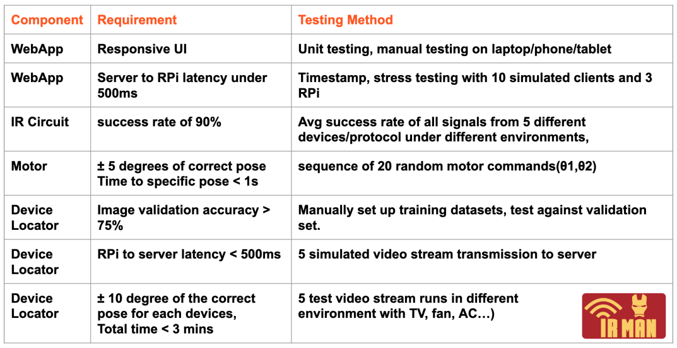


Fig. 11: IR Man SolidWorks CAD

V. Metric and Validation

For testing our components and the final product against the our requirements, we designed the following test and validation methods referring to the metrics in the following table.



Component	Requirement	Testing Method
WebApp	Responsive UI	Unit testing, manual testing on laptop/phone/tablet
WebApp	Server to RPi latency under 500ms	Timestamp, stress testing with 10 simulated clients and 3 RPi
IR Circuit	success rate of 90%	Avg success rate of all signals from 5 different devices/protocol under different environments,
Motor	± 5 degrees of correct pose Time to specific pose < 1s	sequence of 20 random motor commands(θ1,θ2)
Device Locator	Image validation accuracy > 75%	Manually set up training datasets, test against validation set.
Device Locator	RPi to server latency < 500ms	5 simulated video stream transmission to server
Device Locator	± 10 degree of the correct pose for each devices, Total time < 3 mins	5 test video stream runs in different environment with TV, fan, AC...)

Fig. 12: IR Man Metrics and Testing

A. Component Testing

1) Webapp server: We plan to test the webapp server by unit testing by seeding the database with simulated users and devices. To test the responsiveness of the user interface, we plan to manually test the webapp. Another approach is to use UI testing framework, however, since our webapp is simple enough to be manually tested, automated UI testing framework seems like an overkill. For the latency between the webapp server and RPi, we are planning using timestamp to measure the latency and stress testing the system using 10 simulated users and 3 RPi.

2) Device Locator: For the validation accuracy, we intend to manually set up training dataset of labeled IR devices. After we trained our CV model, we need to test it against our validation set. For the latency of video streaming, we are planning to have 5 simulated video stream transmission to the server and use timestamp to keep track of the total time for all the images to reach the server. For the overall CV pipeline testing, we would set up three different IR devices: TV, fan and AC in the room and measure the total time it takes to find these device locations and the accuracy of the device locations. We would run this test for 5 times, each time moving these devices to a different location and use different background environment.

3) IR circuit: For the IR circuit, we plant to test the success rate of IR signal transmission by taking the average success rate of 5 tests. Each time, we would find 5 different IR devices of different device brands and device types, preferably using different protocols, to make sure that our product supports most of the IR devices currently in the market.

4) Motor Control: We plan to test the accuracy and speed of motor control by generating a sequence of 20 random pose. For each run, the motor needs to be within 5 degrees of the final destination, and the total time that it takes to reach a specific location should be less than 500ms.

B. System Testing

We plan to test our final product against the following user story sequentially:
1. User register new account and IR Man device.
2. User add new IR devices to IR Man.
3. IR man scans the environment, locate all devices location.
4. User gets a list of devices to choose from.
5. User chooses a single device and goes into the device control panel.
6. User press a button on the control panel.
7. IR man point to the device and shoot out the IR signal to device.

Similar to testing the device locator, we would conduct this test 5 times under different backgrounds. Each time we would use three IR devices at different locations.

VI. Design Trade Studies

A. IR Circuit Design

A valid concerns has been raised during our design ideation and the peer review. Some peers mentioned that having a ring of IR emitters to broadcast IR signals instead of having having the IR Man turn to the direction of a specific device. Although a ring of IR emitters would cut down system complexity by a lot, it is at risk of signal cross interference and uniqueness in design. Firstly, a lot of IR controlled devices of the same brand might be controlled with the same IR signal. For example, it might not be user's intention to turn on all of his Dell devices at the same time. Therefore, a ring of IR emitters broadcasting IR signals in all direction would cause unwanted behaviour of the system. Thus, we want to have the IR Man turn to the specific location of the IR device and pinpoint that control signal without affecting other IR devices. Another concern is that having the IR Man drive its motors based on computer vision is a unique solution to the problem and we want to keep it that way since it is an interesting engineering problem to solve with lots of added value like line of sight, product differentiation and interactivity.

B. webapp server to RPi connection

For the connection from the webapp server to various RPi, we chose to use websockets, because websockets enable the server and client to send messages to each other at any time, after a connection is established, without an explicit request by one or the other. This is crucial to our system because we are building a real time system: the webapp server could be sending requests to RPi at any time. In a challenge-response system, there is no way for clients(RPi) to know when new request is available for them. The only similar implementation is RPi polling the webapp server periodically, but that's not exactly what we want. Thus, this connection is established through websocket.

C. Image from RPi to device locator

Since we have requirements for the overall time of finding devices, and that the computer vision inference takes most time, we need to cut down the time it takes to transmit the images. In order to send the video streams from RPi to the device locator server, we need to serialize and de-serialize the image frames. We plan to use ImageZMQ over websockets to achieve faster image transmission.

One thing to note is that the input frame size for the neural network model is not decided yet, in that we are not sure how much we shall trade resolution (possibly related to validation accuracy) for speed. This is to be decided after testing and benchmarking CV pipeline with different input frame sizes.

D. Webapp Server and Device Locator Server

Instead of one server that handles everything, we plan to have two servers: the webapp server and the device locator server. We choose to separate these two servers because they have fundamentally different functions and there is no communication between these two servers. This is also a scalable design, once the system grows larger, we can just scale these two kinds of servers separately.

E. Front End User Interface

We choose to develop our front end user interface on a webapp instead of an iOS or Android app because we want to have a platform independent solution. Although a webapp possesses a lot of limitation in terms of performance and feature access through user's native device, and might not provide the best possible user experience, we still choose this to be our approach for the final front end user interface because it is relatively easy and fast to deploy. Given the limited time and resources we have for this project, the best way, under our careful consideration, to support all the user features for the largest audience is through a webapp.

F. Robotics Motion & Dynamics

We decided to incorporate the robotics motion and dynamics feature in this project early on in ideation phase mainly for two reasons. There has been found lot of universal IR devices on the consumer market, and when we were asked the question, ”what makes your project different from that of the others”, we immediately delved into a brainstorm process for product differentiation. Since our project has to do with IR signals, and all of the team members are Marvel's fan, we thought of using Iron Man's figurine as our chassis and re-engineer Iron Man's gauntlet glove into a IR emitter device, and call it the ”IR MAN” instead. The clever pun led us into an elaboration of robotics motion and dynamics. Some peers have questioned the purpose and given us feedback about the robotics part. There are indeed many ways to place the IR diodes, namely put a ring of IR diodes on the chassis to send out the same IR signal in all directions, but we decided to put it in the center of IR MAN's gauntlet and plan motions accordingly to aim the IR emitter directly at the target device. This would solve the line of sight issue, as well as reducing power consumption of the system. This robotics approach adds another layer of computer vision and dynamics control complexity to the project. However, not only does it make IR MAN unique, but also makes it more fun to interact with. In addition, the decision to use a servo as opposed to a second stepper motor is because of weight concerns. The servo is relatively light and can be easily mounted on the body of IR Man.

G. Image Pre-processing

Knowing that the input of camera feed is fixed at 1080p (1920×1080 pixels), and that the input for image inference does not require resolution at this level, we decide to divide the image pre-processing phase into 2 sections, and perform them on different platform. Pre-process of a frame before inference typically consists of resizing the image and normalizing the channel values. The previous phase shrinks the size of an image while the latter part expands the size by changing the data size of color channel from int8_t (a byte) to float (8 bytes typically). Therefore, knowing that all frames are to be transferred from RPi to a remote server, we decide to run image resizing on RPi before image transfer, and image normalization on remote server, such that the image to be transferred is in its smallest size, and the transferred speed maximized under limited network bandwidth.

H. Computer Vision & Neural Network Framework

Due to the limited computing power on the RPi, we decided to run our computer vision module on a remote server. The object detector we chose is YOLOv3. Reason that we pick this model out of many others is that it balances accuracy and inference fps better than the other popular models.

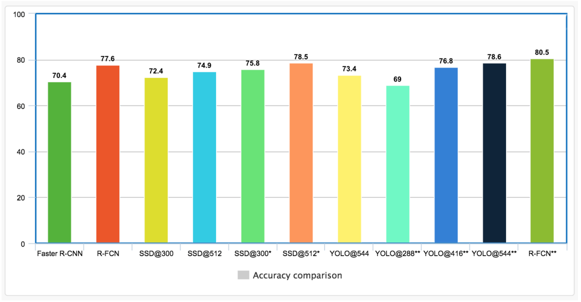


Fig. 13: Accuracy of popular models compared upon PASCAL VOC 2007 and 2012

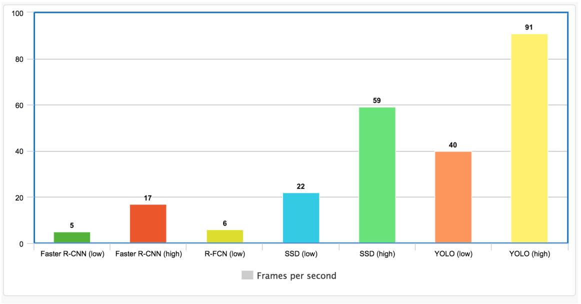


Fig. 14: Inference FPS of popular models compared upon PASCAL VOC 2007 and 2012

It is shown in the graph that validation accuracy doesn't deviate much between models, but YOLOv3 has a large space between its upper and lower limit of inference speed, which means we would be able to tune our image size and control the trade off between speed and accuracy.

VII. Project Management

A. Schedule

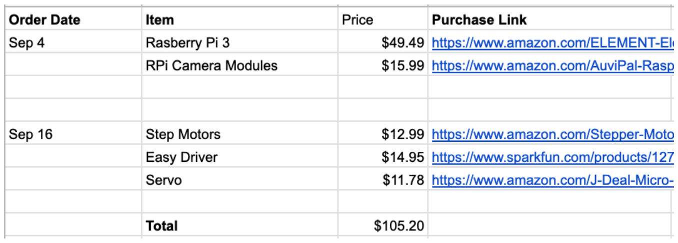
Refer to Gantt chart is in Appendix A.

B. Team Member Responsibilities

Max is responsible for doing all the web application design and deployment. Jiaqi is responsible for doing all the hardware design and implementation. Shirley is responsible for doing all the Computer vision related design and development, as well as server side of object detection. This is a crude assignment, and during development process, teammates will help out each other if necessary.

C. Budget & Bill of Materials

Here is our current bill of material excluding rentals fees of AWS EC2 GPU instances.
The most updated bill of materials can be accessed through our Google Spreadsheets [3].



Order Date	Item	Price	Purchase Link
Sep 4	Rasberry Pi 3	$49.49	https://www.amazon.com/ELEMENT-El
	RPi Camera Modules	$15.99	https://www.amazon.com/AuviPal-Rasp
Sep 16	Step Motors	$12.99	https://www.amazon.com/Stepper-Moto
	Easy Driver	$14.95	https://www.sparkfun.com/products/127
	Servo	$11.78	https://www.amazon.com/J-Deal-Micro-
	Total	$105.20	

Fig. 15: Bill of Materials

D. Risk Management

1) Camera stabilization: When the robot spins around sending video stream to device locator, the camera may suffer from jitter and turbulence. If so, we need to change our implementation: let the robot turn for a certain degree, take a photo after it stabilizes and then turn again...

2) Websocket connection: The current design uses a single websocket as the connection between the webapp server and all the RPis. This might not be an issue, as the requests sent to the RPi are limited in size. However, The current design also uses a single websocket to handle video streaming from all RPis to the device locator server. Thus, if we have more than a few RPis sending video stream to the server at the same time, latency could become an issue. Thus, we need to find a more scalable solution if necessary.

3) Device Locator Usability: The current design uses computer vision to identify the IR devices in the surroundings. It finds the location for TV, AC, fan by object detection. However, in case where the CV methods are not ideal, then we plan on letting the user directly control the robot to point to the devices through the webapp. This meant that we need to change the communication protocol between webapp server and RPi and supporting video streaming on the webapp server.

4) Budget Constraint Risk: We think that budget is a risk factor because $600 is allowed for the entire project and any replacement for broken parts would cause major problems in proceeding the project from moving forward. To mitigate this risk, we decided to reuse any of the parts in the ECE inventory before acquiring them externally.

5) Test Device Risk: We are planning to find 5 IR devices to test our product. It could be difficult to find these devices to test. We currently have an AC, two TVs and a lamp. If we are unable to find enough IR devices to test, we might need to borrow or buy some low cost test devices: speaker, fans, etc.

Appendix A
IR MAN Gantt Chart

Refer to Figure 16 for the Gantt Chart.

Acknowledgment

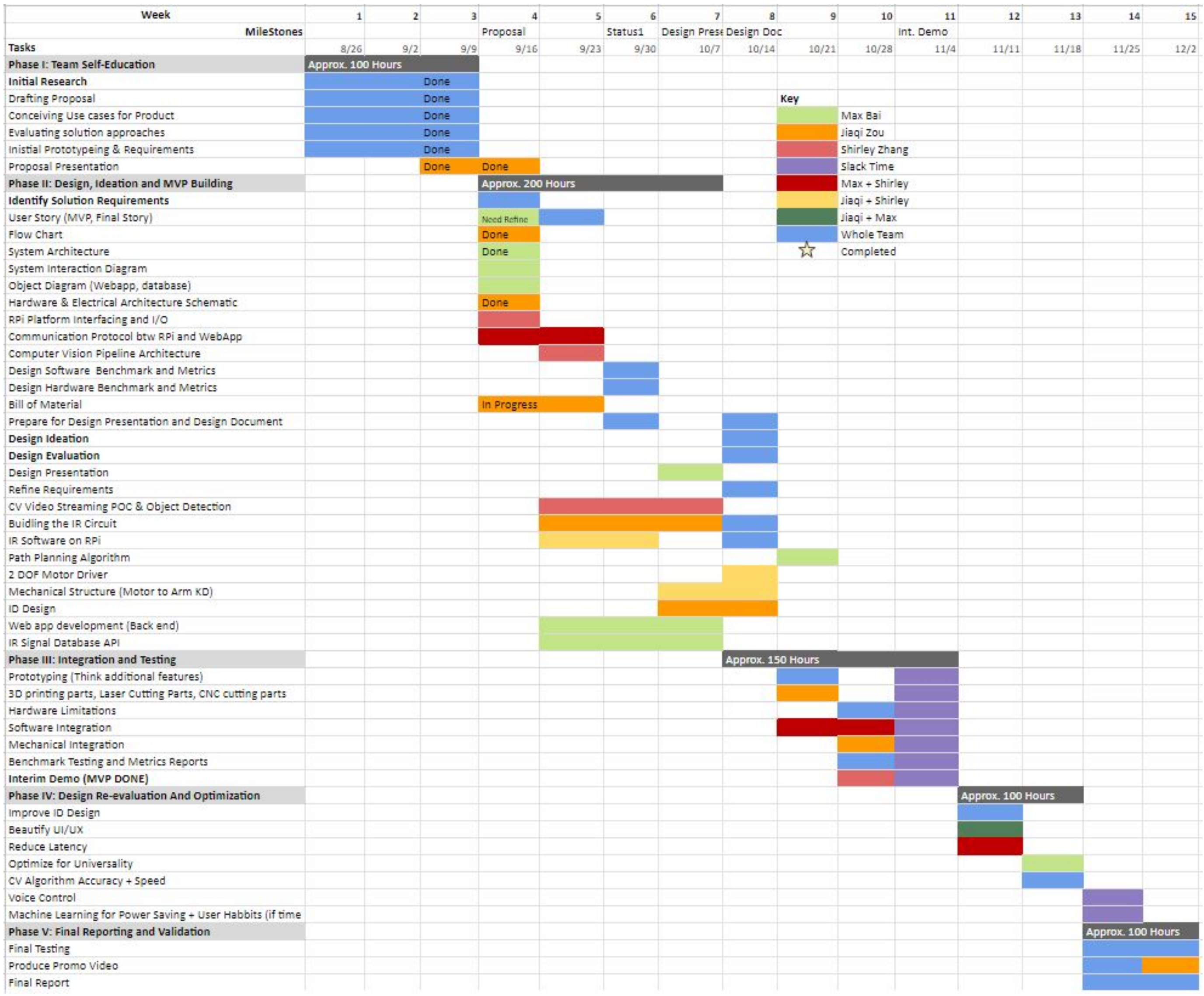
Thanks CMU ECE department for giving us the opportunity to work on a capstone project and transform our knowledge into real world applications. Thanks Prof. Bill Nace for providing the course logistics and framework for us to work efficiently on our project. Thanks Prof. Vyas Sekar and Prof. Shawn Kelly for providing us lots of technical support and advice during our development. Thanks Ranganath (Bujji) Selagamesetty for providing us technical supports and logistical supports and being a good TA/friend.

References

[1] LIRC: Linux Infrared Remote Control Webpage
`http://lirc.sourceforge.net/remotes/`

[2] Jonathan Hui: Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and YOLOv3)
`https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359`

[3] Google Spreadsheet: Bill of Material for IR Man)
`https://docs.google.com/spreadsheets/d/12K-8_Rmk1GgIzw4O3nj8dHEQQZHdu0VqMjXnk-BmZIE/edit?usp=sharing`



Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MileStones				Proposal		Status1	Design Prese	Design Doc			Int. Demo				
Tasks	8/26	9/2	9/9	9/16	9/23	9/30	10/7	10/14	10/21	10/28	11/4	11/11	11/18	11/25	12/2
Phase I: Team Self-Education	Approx. 100 Hours														
Initial Research	🟦	🟦	🟦 Done												
Drafting Proposal	🟦	🟦	🟦 Done						**Key**						
Conceiving Use cases for Product	🟦	🟦	🟦 Done						🟩	Max Bai					
Evaluating solution approaches	🟦	🟦	🟦 Done						🟧	Jiaqi Zou					
Inistial Prototypeing & Requirements	🟦	🟦	🟦 Done						🟥	Shirley Zhang					
Proposal Presentation			🟧 Done	🟧 Done					🟪	Slack Time					
Phase II: Design, Ideation and MVP Building				Approx. 200 Hours					🟫	Max + Shirley					
Identify Solution Requirements				🟦					🟨	Jiaqi + Shirley					
User Story (MVP, Final Story)				🟩 Need Refine	🟦				▦	Jiaqi + Max					
Flow Chart				🟧 Done					🟦	Whole Team					
System Architecture				🟩 Done					☆	Completed					
System Interaction Diagram				🟩											
Object Diagram (Webapp, database)				🟩											
Hardware & Electrical Architecture Schematic				🟧 Done											
RPi Platform Interfacing and I/O				🟥											
Communication Protocol btw RPi and WebApp				🟫	🟫										
Computer Vision Pipeline Architecture					🟥										
Design Software Benchmark and Metrics						🟦									
Design Hardware Benchmark and Metrics						🟦									
Bill of Material				🟧 In Progress	🟧										
Prepare for Design Presentation and Design Document						🟦		🟦							
Design Ideation								🟦							
Design Evaluation								🟦							
Design Presentation							🟩								
Refine Requirements								🟦							
CV Video Streaming POC & Object Detection					🟥	🟥	🟥								
Buidling the IR Circuit					🟧	🟧	🟧	🟦							
IR Software on RPi					🟨	🟨		🟦							
Path Planning Algorithm									🟩						
2 DOF Motor Driver								🟨							
Mechanical Structure (Motor to Arm KD)							🟨	🟨							
ID Design							🟧	🟧							
Web app development (Back end)					🟩	🟩	🟩								
IR Signal Database API					🟩	🟩	🟩								
Phase III: Integration and Testing								Approx. 150 Hours							
Prototyping (Think additional features)									🟦		🟪				
3D printing parts, Laser Cutting Parts, CNC cutting parts									🟧		🟪				
Hardware Limitations										🟦	🟪				
Software Integration									🟫	🟫	🟪				
Mechanical Integration										🟧	🟪				
Benchmark Testing and Metrics Reports										🟦	🟪				
Interim Demo (MVP DONE)										🟥	🟪				
Phase IV: Design Re-evaluation And Optimization												Approx. 100 Hours			
Improve ID Design												🟦			
Beautify UI/UX												▦			
Reduce Latency												🟫			
Optimize for Universality													🟩		
CV Algorithm Accuracy + Speed													🟦		
Voice Control														🟪	
Machine Learning for Power Saving + User Habbits (if time														🟪	
Phase V: Final Reporting and Validation														Approx. 100 Hours	
Final Testing														🟦	🟦
Produce Promo Video														🟦	🟧
Final Report														🟦	🟦

Fig. 16: Gantt Chart for IR Man