AMITY UNIVERSITY

STUDY & DEVELOPMENT OF VOICE CONTROLLED WHEELCHAIR BY IOT

Submitted by	guided by
Mr. Abdul Rahoof	Prof.(Dr.) Neeraj Kumar
A2327018006	Mr. Bhupendra Singh
M Tech Mechatronics	
ASET Noida	

Index

Acknowledgement

I would like to express my gratitude to **Prof.(Dr.) Neeraj Kumar** and **Mr. Bhupendra Singh** for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project. I would also like to extend my sincere thanks to **AMITY UNIVERSITY**

1. Abstract

As we, all know there are lots of disabled peoples in our worlds and the health service sector is making continuous efforts to improve the service to make easy mobility for these peoples. In the growing technology, many developers are coming up with different ideas of robotic wheelchairs. These are wheelchairs, which can easily direct the patient from one place to another by simple means of controlling. They have sensors embedded in them to navigate and detect the necessary obstacles. Here I have developed an intelligent wheelchair, which can be controlled by the voice command of the patient and can be easily directed to the desired place. Unlike other voice command system, which uses simple voice module, I have used the feature of IOT to enhance the functionality of the wheelchair. It consist of a raspberry pi, dc motors, relays, and the use of IFTTT server. IFTTT is used to make the trigger and response applets, which will help the microcontroller to analyse which code is to be executed at the given time.

2. Introduction

There are many disable people in this world, and many of them do not have proper functioning limbs. Many of these peoples are bounded themselves to a wheelchair for the purpose of mobility. Some of them can move there wheelchair by the mere application of force on the wheels and some wheelchairs come with remote controlled functionality. However, many patients do not even have proper working arms. Those peoples are always depended upon others to make them move from one place to another.

This project is about making a wheelchair, which can be controlled by simple voice command. It can help a disabled person who cannot move or have their limbs to control the wheelchair to go around just by giving simple voice commands. These command will be very simple like forward, backward, stop etc. this functionality will give the person a new experience to operate his wheelchair and make it possible for them to move freely from one place to another by their own. The voice commands will be given to the wheelchair just by speaking in the mic of their mobile phone or any device they have which is been integrated with google voice assistant. This is done by using IFTTT and interfacing it with the microcontroller so that it can receive instructions via internet and the microcontroller is then interfaced with the motors that will be connected to the wheels of the wheelchair, which will result in the desired movement of the wheelchair.

3. Methodology

A main control unit will control the wheelchair. The main control unit consist of a raspberry pi 4. It is been integrated with a motor shield drive so that the motors can be controlled in both the directions without altering the connections. Then the commands has to be prepared which is been made possible by making applets in the IFTTT. This applets works as trigger packets. When a specific command is given, it triggers the applet, which sends a response to the IOT server, which will be received by our microcontroller. These applets will work on the basis of coding we have made in a platform called Particle.io. This platform can be easily integrated with the raspberry pi and the codes can be flashed in the microcontroller very easily. So a program is been written in the particle.io web ID and is been trigger set by the applets. So when a command is been given it will see the response code for that specific command and will instruct the microcontroller to initiate that code. This initiation of code will help the rotation of motor in specific direction, which will in turn make the wheelchair move. These codes can be initiated by giving the command in the user's mobile phone itself, which makes the functionality for user easier.

4. Literature review

Nowadays there are lots of study research on the field of remotely controlled wheelchairs. Many pioneers have been doing their research to how to make a more suitable wheelchair, which can be easily controlled by the disabled peoples. Once there was an age when there use to be a study of mechanical or conventional wheelchair. Then there comes the age of electronics and researchers starts working on the development of remote controlled wheelchair. Now is the era of IOT and artificial intelligent where we are researching on the working of robotic wheelchair.

Rafael Barea, Luciano Boquete, Manuel Mazo[1] developed a wheelchair for disabled patients which can be controlled by the movement of the their eye by means of electrooculography. It works on the principle of capturing electrooculograms by an acquisition system. Then the wavelet and the neural networks are analysed in real time. It had a disadvantage in working under dim lighting and sometime causes itching sensation after a long use.

Nathalia Peixoto , Hossein Ghaffari Nik, Hamid Charkhkar developed a wheelchair which can be controlled by the humming of the patient. Here they used an accelerometer measuring vocal cord vibration. The speed of the wheelchair is controlled by the frequency of the hum produced.

N. Shinde and K. George [12] designed a robotic wheelchair, which can be controlled and driven by brain waves produced by the user, and eye blinks of the user. The disadvantage of this was that they need to be calibrated every time because the brain impulse varies from person to person.

5. Approach

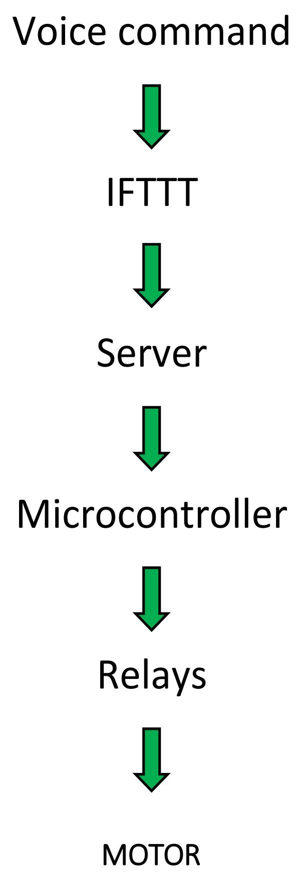


Fig 1: Basic flow chart of the process

- Voice command is given through a mic (any device can be used). The voice command is been modulated and been send to the IFTTT.
- IFTTT is a platform where we can set various actions to perform by activating certain trigger. This will be done in an appropriate server where the codes are been written.
- Now the instruction coming from the server is been received by the module and is sent to the microcontroller.
- Character by character from the serial buffer which is been sent by the app is accepted and combine them to form a string. This string together forms the command.
- When the proper command is received to the microcontroller, then the motor rotates in a specific manner according to the given command.
- This will result in forward, backward and turning motion of the wheelchair.

6. Block diagram

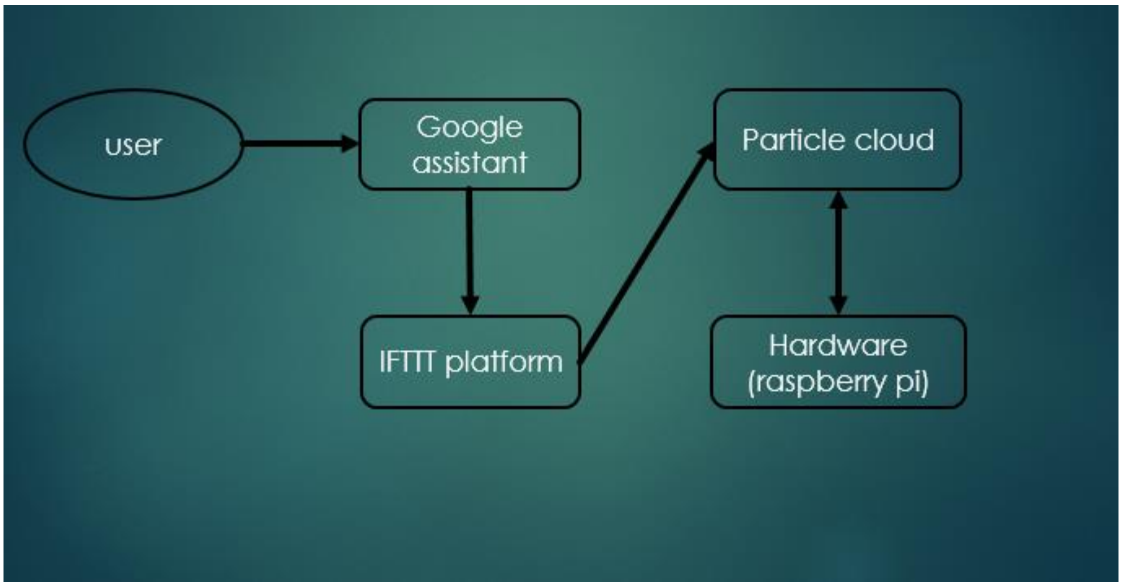


Fig 2: Block diagram of signal transfer

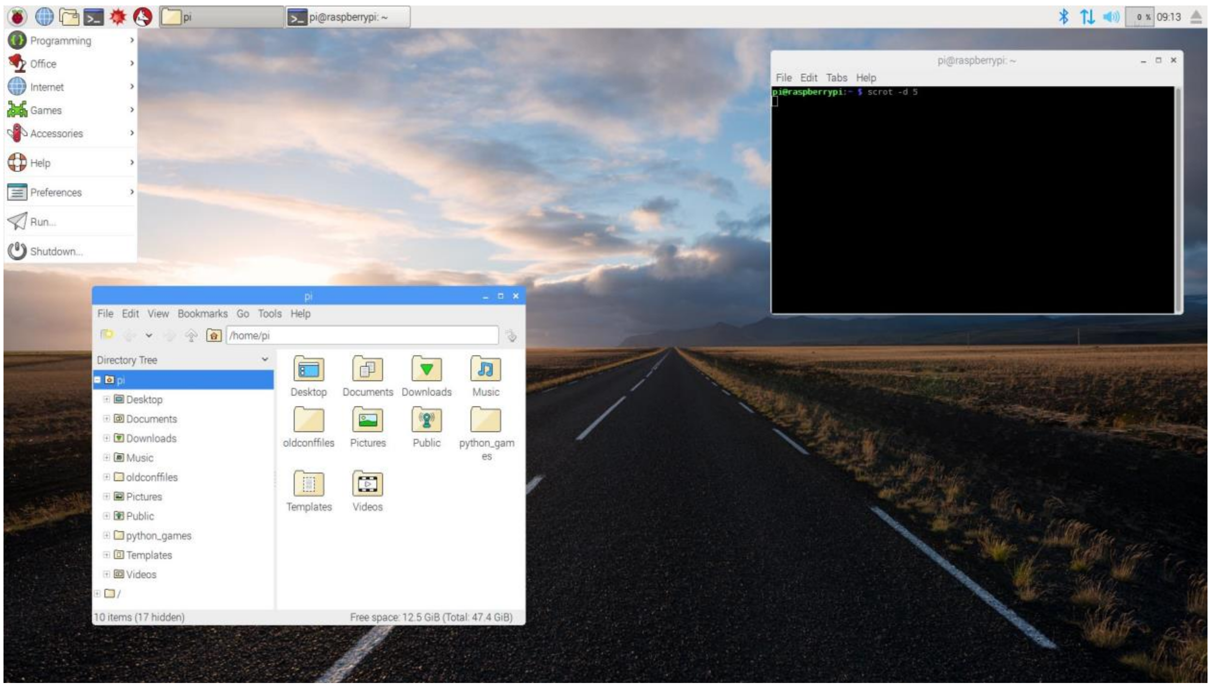
The above fig shows the block diagram of the process of the transmission of the command. As we can see the very first block contain user. It is where the user in a google-integrated device, such as mobile or google home gives the command. This command will be analyzed and synthesize by the google assistant and is been send to the IFTTT platform. This is where our created applet going to get the trigger and will provide the respective response.

Now the response given by the applets is been send to the particle server where we have already made our desired code for the movement of the motors. When the proper trigger response is been received by the particle cloud it subscribe to the following event and publish the code under the given event. When the code is been executed it directly interface with the raspberry pi and this results in the rotation of the motors in the desired direction.

7. Setting up of raspberry pi

Raspberry pi 4 is a microcontroller, which will be the brain of our project. The first thing is to set up the raspberry pi for further use. First, we have to install a desired operating system on the raspberry pi. In my case, I have installed raspbian operating system. It can be easily downloaded from raspberry pi official website.

Once the OS is downloaded copy it in a micro SD card and the card is to be inserted in the SD card slot in the raspberry pi. Then connect the pi with a display device to work as a monitor Any HDMI/DVI monitor or TV should work as a display for the Pi. Switch on the pi power button. The Pi is powered by a USB Type-C [model 4B]. Once the power is on the OS installation window will open up. Just follow the basic installation procedure and it will install a raspbian OS in the pi. Once it is done the pi desktop will open which will look similar to our windows desktops with slight difference. Raspbian OS comes up with many preinstalled applications for making the working experience more flexible in the pi.



```
pi@raspberrypi:~ $ scrot -d 5
```

Fig 3: Start up window of raspbian OS

7.1. Particle.io

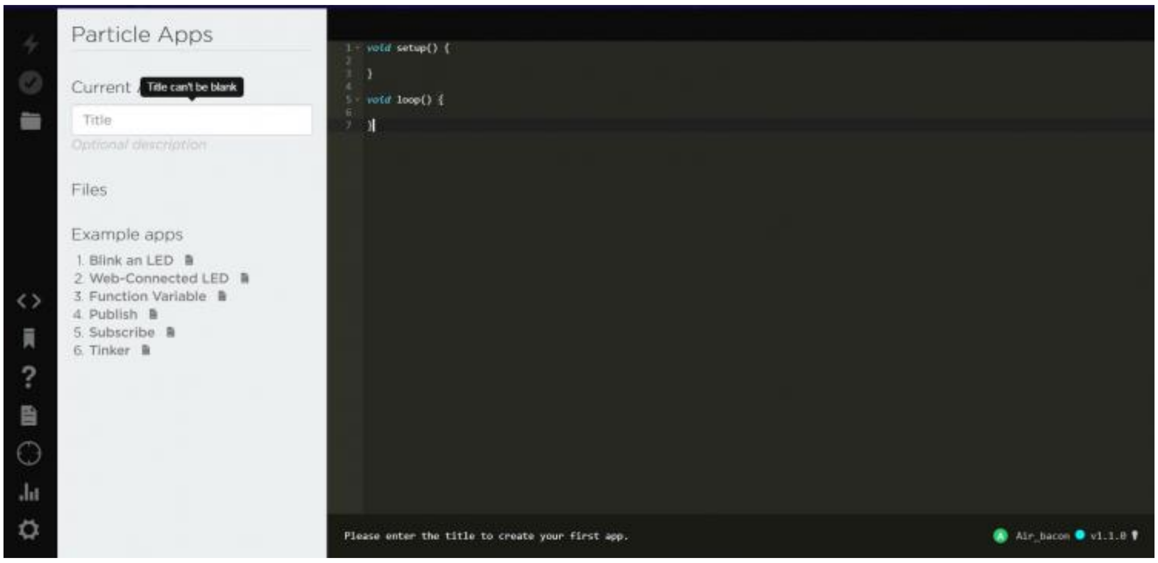
Particle Build is an Integrated Development Environment, or IDE. Here we can do software development and necessary coding in an easy-to-use application. It has features to connect to the device we are using, which here is the raspberry pi.

Steps overview:

- Start with opening particle.io website from browser.
- Create a particle account.
- Go to particle web IDE option from the menu.
- It will open the workbench or the main window where all works has to be done.
- Once the device is been connected to the particle then it will flash at the bottom of the workbench.
- Now any code can be written in the browser and can be flashed directly to the raspberry pi.

7.2. Connecting raspberry pie to particle.io

- Download raspbian operating system into the raspberry pi as per the procedure.
- After all starting setup is done, we have to open terminal in raspberry desktop. Terminal is the main window in raspbian where all instructions are given for the working of raspberry pi. It is just like the command prompt of windows.
- Now we have to connect our particle account with our raspberry pi. It is done by entering "bash <(curl -sL https://particle.io/install-pi)" code in the terminal. This command will start to execute a list of function and according to the give credentials of the particle.io. When this step is over it will automatically connect our raspberry pi with the particle account and that can be seen in lower right corner of the particle IDE page. Once the device is been connected to the particle account then we can flash any command from particle to our microcontroller from anywhere on planet.



```
void setup() {

}

void loop() {

}
```

```
Please enter the title to create your first app.
```

Fig 4: Particle web IDE interface

- The above figure shows the overview of the particle IDE where we will be writing our codes and will be flashed to the connected device. Once the device is been flashed it is denoted in green colour in the bottom right of the window.

8. IFTTT (if this than that)

IFTTT is a web-based service, which is used to trigger a response for the proper trigger given to it. This is achieved by the creation of small condition statements called applets. Applets consist of a trigger condition and a response condition. Like if we select a trigger for like when a mail is received in Gmail and a response for turn on the home automation light. Then whenever a mail is been received it will trigger the response and the light will be switched on.

The very first step here is to log into the IFTTT official website and make an account . to do that we have to select the sign in option in the top right corner of the website. It will open a window to fill with required user credentials.

Once the account is been made log in into the IFTTT account. There we can see many options, go for create option in the menu that will open a search window.

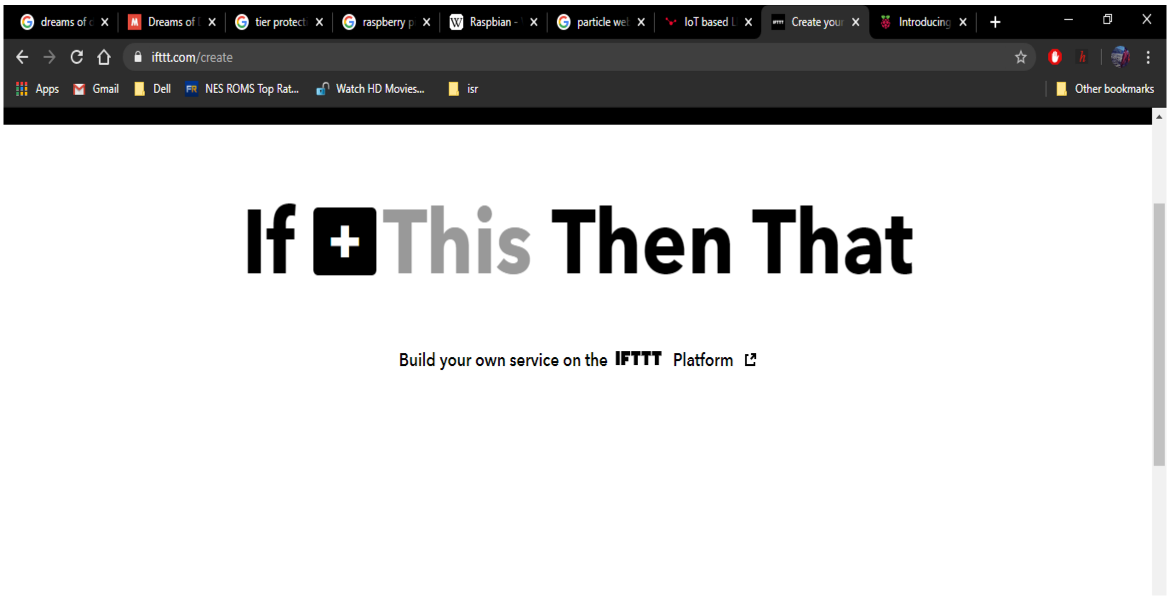


Fig 5: Starting interface of IFTTT

Here we can see a window opens which will be having a highlight over the THIS with a + symbol. We have to click that and it will open a search window with many services. There we can see a search button where we can search for desire service out of many.

8.1. Applet creation

The next step comes up is the applet creation. Here we have to search for google assistant option, which will be the trigger service for our project. Google assistant, which is integrated in our mobile phone, will receive the command and will carry forward it to the particle web service.

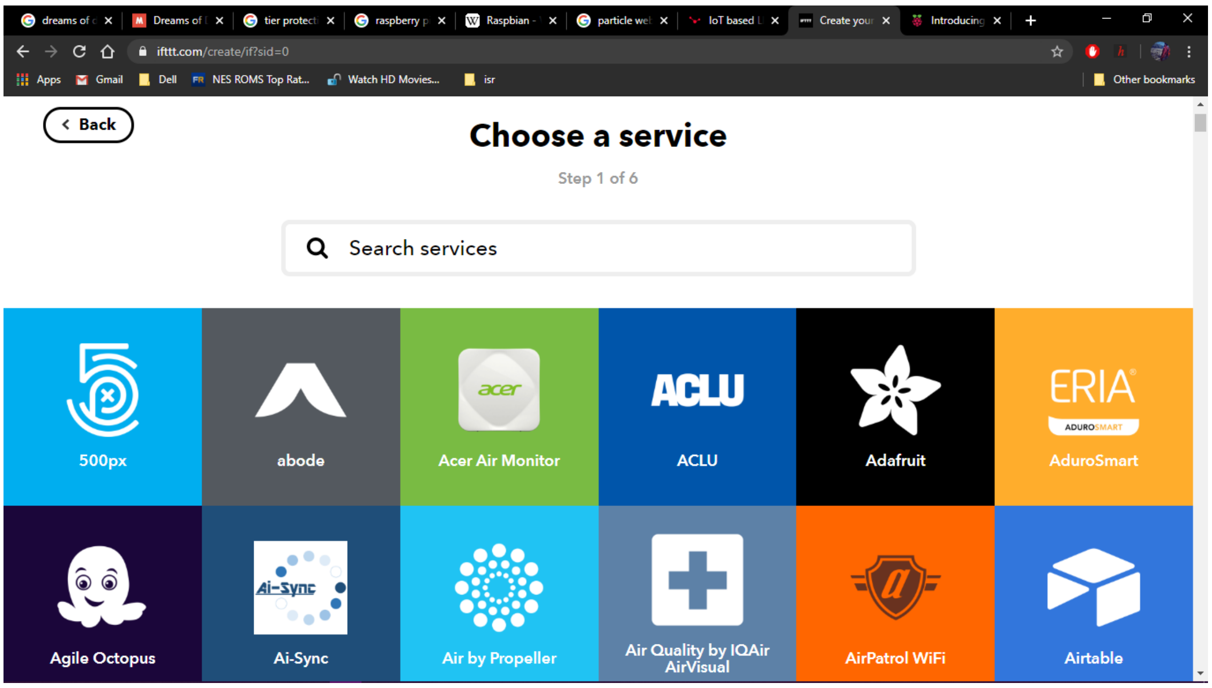
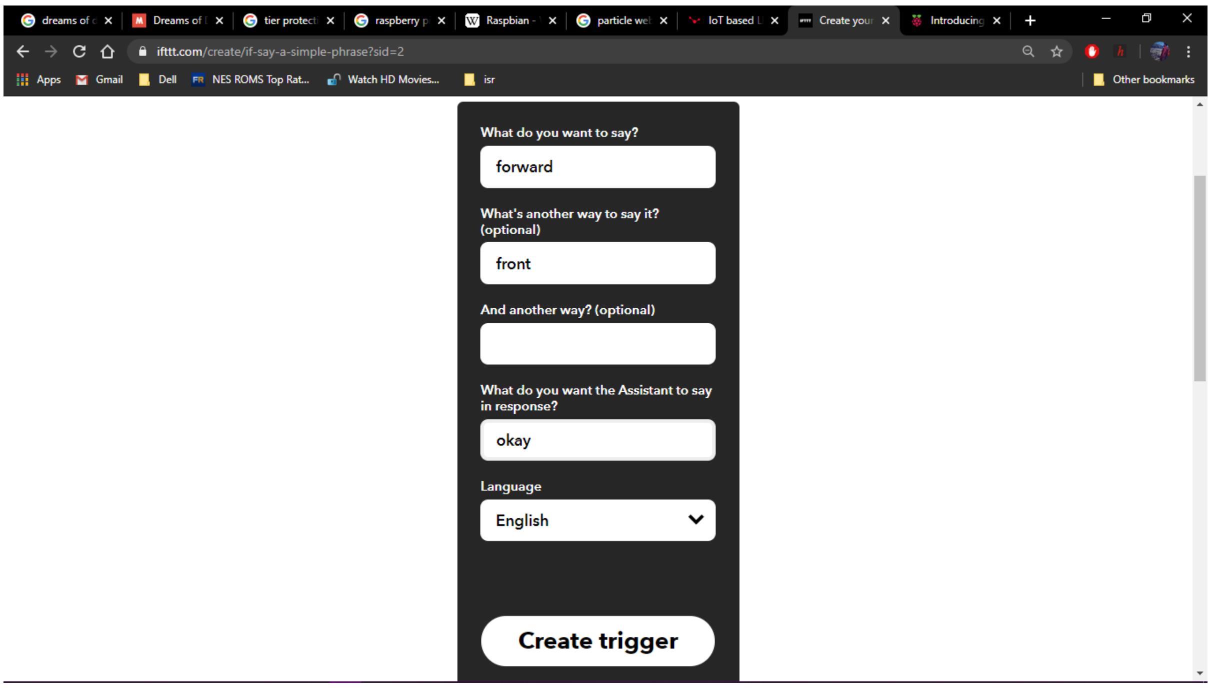


Fig 6: Service selection page

. Once that is been done we have to repeat the same procedure for the THAT option. In that option we have to search for the particle option which will be our response service.


What do you want to say?

forward

What's another way to say it? (optional)

front

And another way? (optional)

What do you want the Assistant to say in response?

okay

Language

English

Fig 7: Command selection for google assistant

Once the google assistant is been selected it will open a window asking for the command we want to say to the assistant. Here our first applet is for the forward command so we put forward command in the required place and select what the assistant should answer back to verify the command. Then press the create trigger button.

After that the same previous window will open with THAT as the highlighted one. Click on that button and search for the particle option. Once that is been done it will ask for what type of service we are expecting for. There we have to select the publish event option.

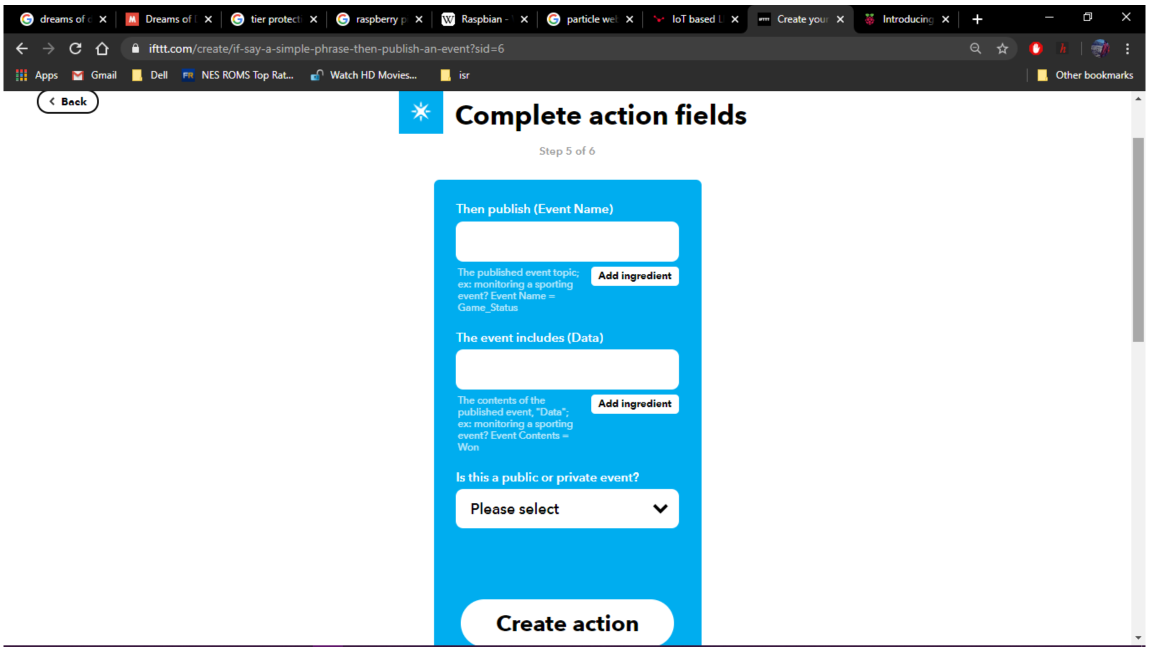

Back
Complete action fields
Step 5 of 6
Then publish (Event Name)
The published event topic; ex: monitoring a sporting event? Event Name = Game_Status
Add ingredient
The event includes (Data)
The contents of the published event, "Data"; ex: monitoring a sporting event? Event Contents = Won
Add ingredient
Is this a public or private event?
Please select
Create action

Fig 8: Event subscription setup page

In this window we have to give our event a proper name which in our case will be "move forward" this event name should match the code event name so that the service can get to know which code it need to run in order to achieve the required motion of the wheelchair.

Once these steps are done, our one of the applet is created. For each command, we have to create that many applets. In our case, we will be creating five applets for moving forward, backward, left, right, and stop.

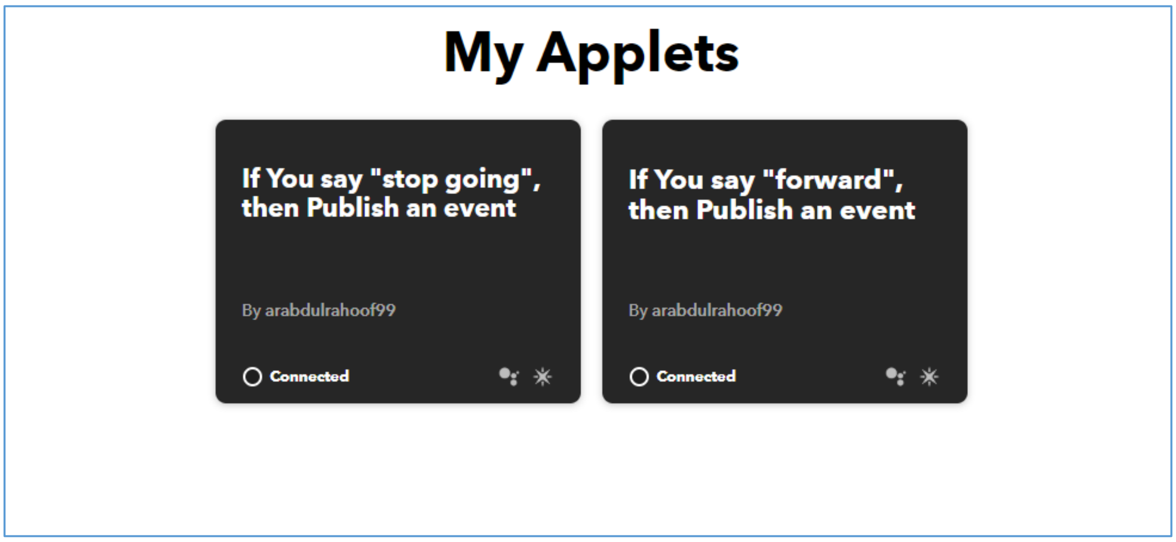

My Applets
If You say "stop going", then Publish an event
By arabdulrahoof99
Connected
If You say "forward", then Publish an event
By arabdulrahoof99
Connected

Fig 9: created applets

9. Motor interfacing

Once the software setup is done then comes the setting up of hardware. The thing we have to achieve now is to interface our raspberry pi with the motors. Here we will be using two dc motors each with specification of 9v and 300 rpm. However, here we cannot directly connect the motor with the pi. That is because here we need the bidirectional rotation of the motor as per the requirements and for doing so we need to change the wiring of the motor, which will be difficult to achieve once the circuit is complete. So to achieve this will be using a motor drive shield between the connection of motor and the pi.

A motor drive shield or H-bridge is a chip, which allows the voltage to be apply on both terminals in both directions so that there will be no need for rewiring and we can get bidirectional rotation of the motor whenever we need. Here we will be using L293D motor drive. It is a 16 pin IC designed to drive up to 4 bidirectional DC motors with an individual 8-bit speed selection. It contains 4 H-bridges which provide up to 0.6 A per bridge (1.2A peak) at voltages from 4.5 V to 36 V.

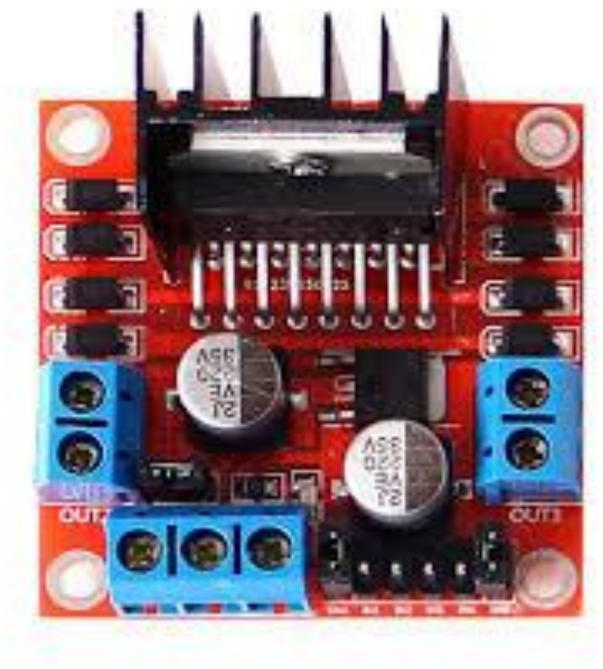
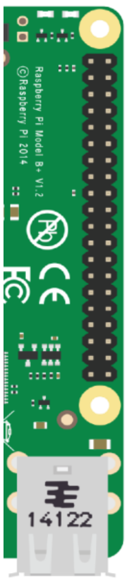
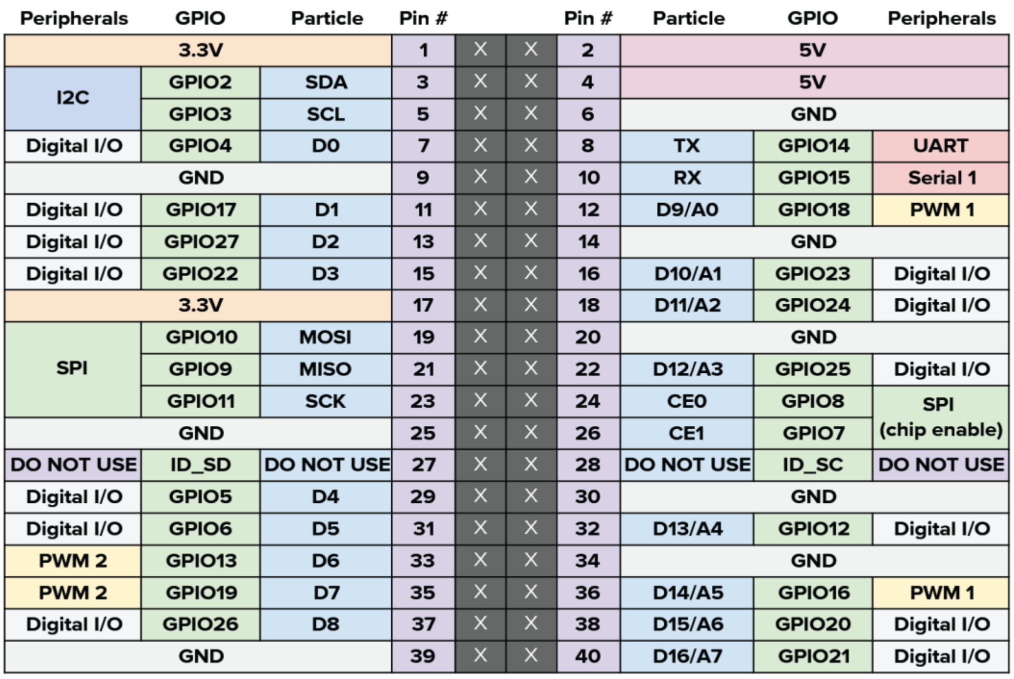


Fig 10: L293D motor driver

10. GPIO pinout diagram





Peripherals	GPIO	Particle	Pin #			Pin #	Particle	GPIO	Peripherals
	3.3V		1	X	X	2		5V	
I2C	GPIO2	SDA	3	X	X	4		5V	
	GPIO3	SCL	5	X	X	6		GND	
Digital I/O	GPIO4	D0	7	X	X	8	TX	GPIO14	UART
	GND		9	X	X	10	RX	GPIO15	Serial 1
Digital I/O	GPIO17	D1	11	X	X	12	D9/A0	GPIO18	PWM 1
Digital I/O	GPIO27	D2	13	X	X	14		GND	
Digital I/O	GPIO22	D3	15	X	X	16	D10/A1	GPIO23	Digital I/O
	3.3V		17	X	X	18	D11/A2	GPIO24	Digital I/O
	GPIO10	MOSI	19	X	X	20		GND	
SPI	GPIO9	MISO	21	X	X	22	D12/A3	GPIO25	Digital I/O
	GPIO11	SCK	23	X	X	24	CE0	GPIO8	SPI (chip enable)
	GND		25	X	X	26	CE1	GPIO7	
DO NOT USE	ID_SD	DO NOT USE	27	X	X	28	DO NOT USE	ID_SC	DO NOT USE
Digital I/O	GPIO5	D4	29	X	X	30		GND	
Digital I/O	GPIO6	D5	31	X	X	32	D13/A4	GPIO12	Digital I/O
PWM 2	GPIO13	D6	33	X	X	34		GND	
PWM 2	GPIO19	D7	35	X	X	36	D14/A5	GPIO16	PWM 1
Digital I/O	GPIO26	D8	37	X	X	38	D15/A6	GPIO20	Digital I/O
	GND		39	X	X	40	D16/A7	GPIO21	Digital I/O

Fig 11: GPIO pinout diagram

The figure illustrates the GPIO pinouts of the raspberry pi 4. Total it consist of 40 pins which has various functions as per the details. Here we can see the pin 9 of the pi is termed as GND, it stands for ground. So the ground wire the motor drive will be connected to it. The GPIO pins are responsible for output voltage provided to any device or actuator. If that pin is kept HIGH the current will flow from the pin and if kept LOW no current flows.

Next to the GPIO tab, we can see a particle tab, here we can see the pin names are mentioned in different manners like D1, D2 etc. these are the pin names that is understand by the particle server. So while making our code we have to denote the GPIO pins of the pi in particle denotation.

11. L298n interfacing with Pi

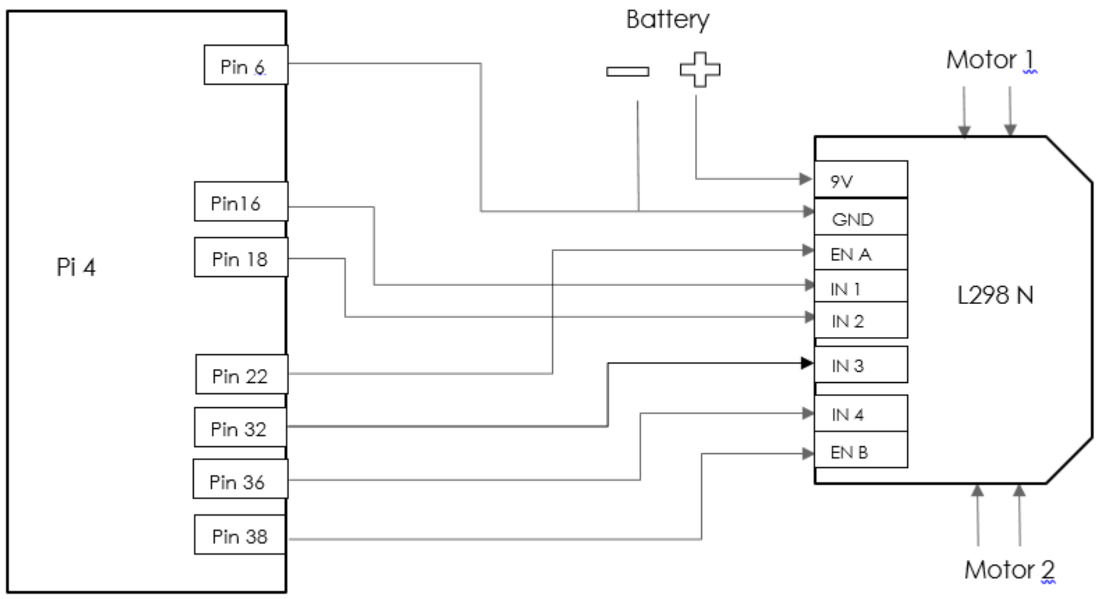


Fig 12: Interfacing of L293D with PI

The fig shows the circuit diagram of the interfacing of the L298n motor shield drive with the PI. As we can see, the L298n consist of 4 output pins for controlling 2 motors at a same time and 8 processing pins which are (voltage, GND, ENA, IN1, IN2, IN3, IN4, ENB). The enable A and enable B pins are responsible for the activation of motor 1 and 2 respectively. The IN1 and IN2 are the two inputs of the motor 1 and IN3 and IN4 are the two inputs of the motor 2.

Now the interfacing is in the following ways:

Pin6 => GND

Pin16 => IN1

Pin18 => IN2

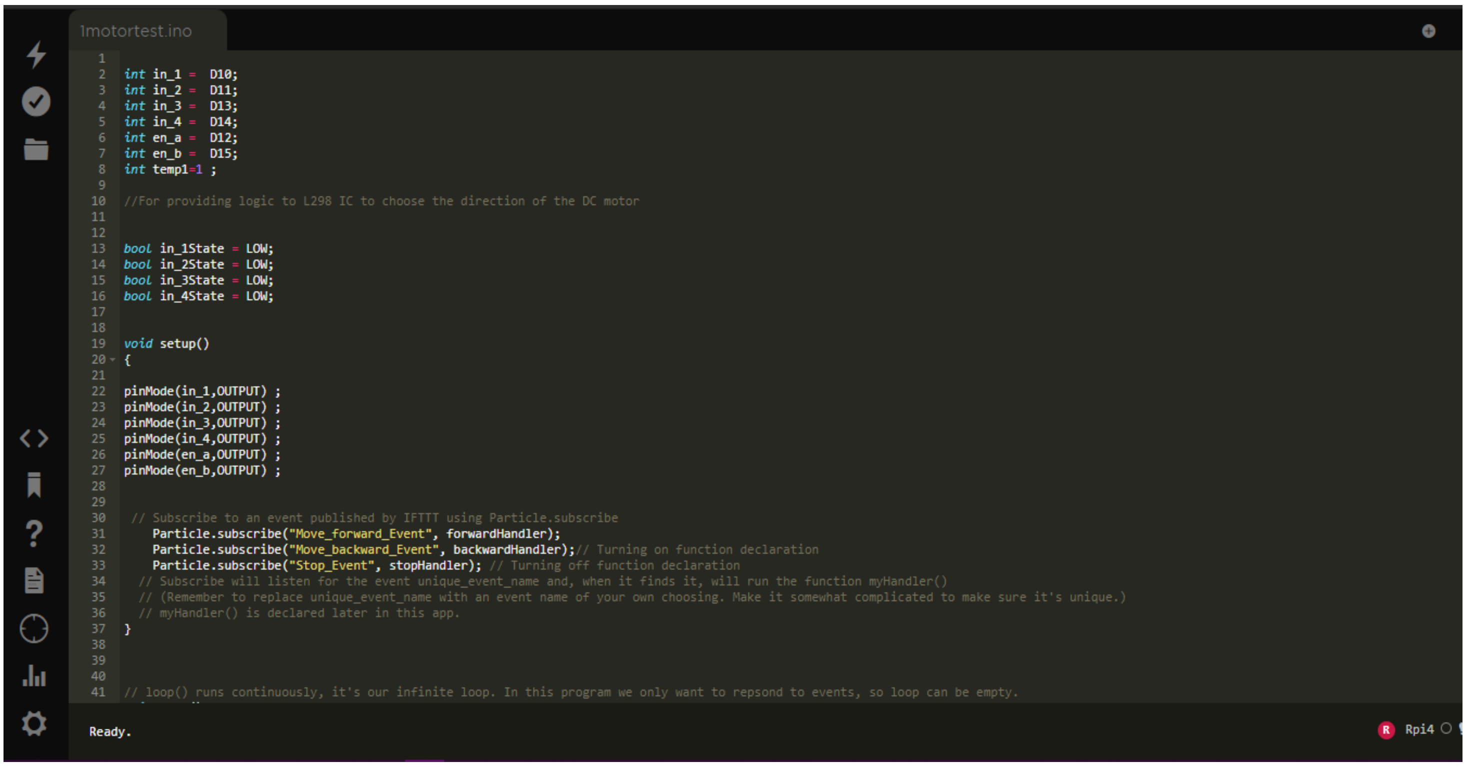
Pin22 => EN A

Pin32 => IN3

Pin36 => IN4

Pin38 => EN B

12. Codes

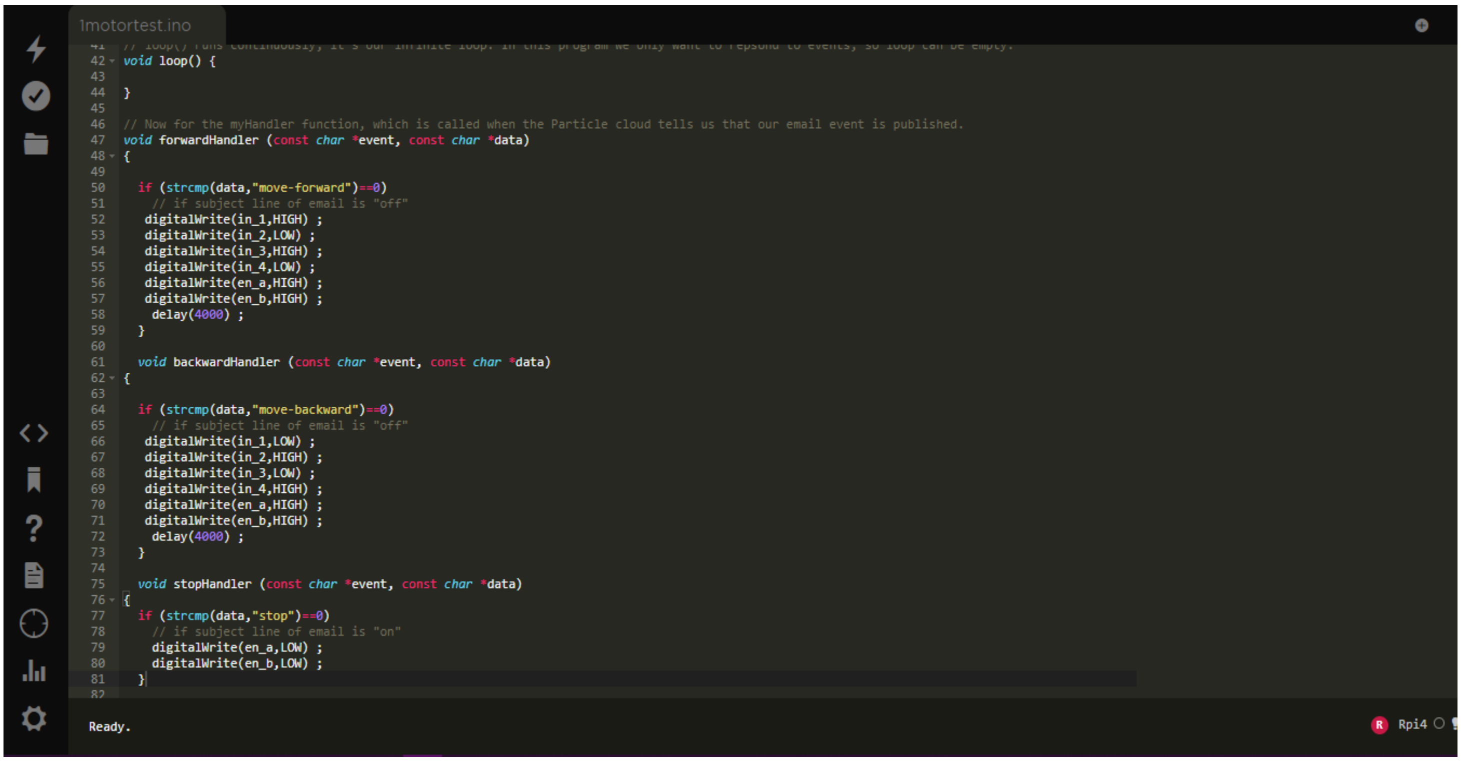


1motortest.ino

```

int in_1 =  D10;
int in_2 =  D11;
int in_3 =  D13;
int in_4 =  D14;
int en_a =  D12;
int en_b =  D15;
int temp1=1 ;

//For providing logic to L298 IC to choose the direction of the DC motor


bool in_1State = LOW;
bool in_2State = LOW;
bool in_3State = LOW;
bool in_4State = LOW;


void setup()
{

pinMode(in_1,OUTPUT) ;
pinMode(in_2,OUTPUT) ;
pinMode(in_3,OUTPUT) ;
pinMode(in_4,OUTPUT) ;
pinMode(en_a,OUTPUT) ;
pinMode(en_b,OUTPUT) ;


 // Subscribe to an event published by IFTTT using Particle.subscribe
    Particle.subscribe("Move_forward_Event", forwardHandler);
    Particle.subscribe("Move_backward_Event", backwardHandler);// Turning on function declaration
    Particle.subscribe("Stop_Event", stopHandler); // Turning off function declaration
  // Subscribe will listen for the event unique_event_name and, when it finds it, will run the function myHandler()
  // (Remember to replace unique_event_name with an event name of your own choosing. Make it somewhat complicated to make sure it's unique.)
  // myHandler() is declared later in this app.
}



// loop() runs continuously, it's our infinite loop. In this program we only want to repsond to events, so loop can be empty.
```

Ready.

Rpi4



1motortest.ino

```
void loop() {

}

// Now for the myHandler function, which is called when the Particle cloud tells us that our email event is published.
void forwardHandler (const char *event, const char *data)
{

  if (strcmp(data,"move-forward")==0)
    // if subject line of email is "off"
   digitalWrite(in_1,HIGH) ;
   digitalWrite(in_2,LOW) ;
   digitalWrite(in_3,HIGH) ;
   digitalWrite(in_4,LOW) ;
   digitalWrite(en_a,HIGH) ;
   digitalWrite(en_b,HIGH) ;
    delay(4000) ;
  }

  void backwardHandler (const char *event, const char *data)
{

  if (strcmp(data,"move-backward")==0)
    // if subject line of email is "off"
   digitalWrite(in_1,LOW) ;
   digitalWrite(in_2,HIGH) ;
   digitalWrite(in_3,LOW) ;
   digitalWrite(in_4,HIGH) ;
   digitalWrite(en_a,HIGH) ;
   digitalWrite(en_b,HIGH) ;
    delay(4000) ;
  }

  void stopHandler (const char *event, const char *data)
{
  if (strcmp(data,"stop")==0)
    // if subject line of email is "on"
    digitalWrite(en_a,LOW) ;
    digitalWrite(en_b,LOW) ;
  }

```

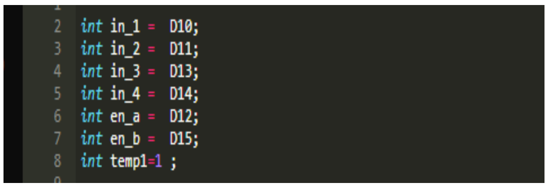
Ready.

Rpi4

Fig 13: Coding for the PI in particle IO

12.1. Code explanation

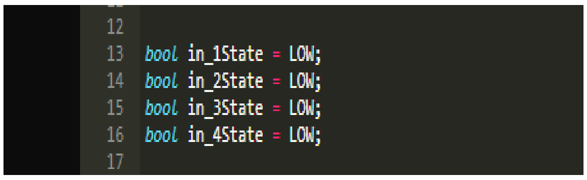
First we will be denoting the pins of the pi in the form of particle denotation and write the proper interfacing pins of the motor drive. This code will help to make the pi understand which pin in connected to which. This is done by giving the ‘int’ data type.



```
int in_1 =  D10;
int in_2 =  D11;
int in_3 =  D13;
int in_4 =  D14;
int en_a =  D12;
int en_b =  D15;
int temp1=1 ;
```

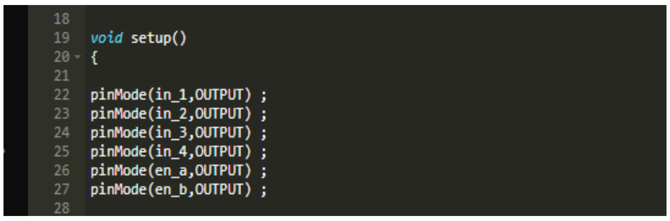
as we can see, here we have given the proper pin interface according to our circuit diagram of pi and L298n. this code has made the pi clear that in_1 is connected to D10 of the pi and so on.

The next code we have written by using the ‘bool’ data type. This is Boolean command and is used for indication that the outcome is always either true or false. In this case it shows the condition of the IN pins of the drive shield and we have kept it in low which tells the pi to keep the current off when the operation is started or when no code is been given.



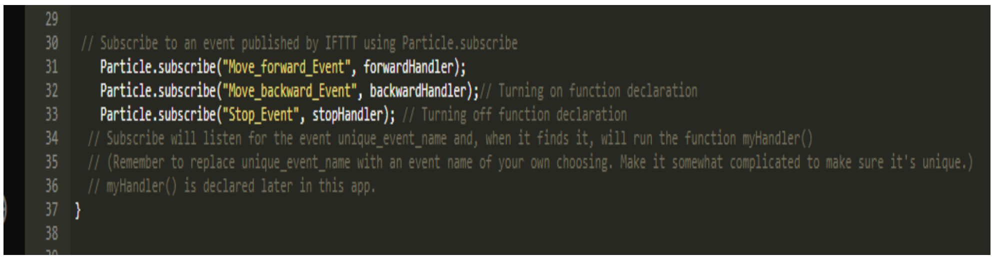
```
bool in_1State = LOW;
bool in_2State = LOW;
bool in_3State = LOW;
bool in_4State = LOW;
```

The next step is to declare which pin are output and which pins are input. In this case all of our 6 pins are output pins because they will be sending the signal out from the raspberry pi to the motor. This is done by the command code ‘pinMode’. So here we have declared all of our pins as output.



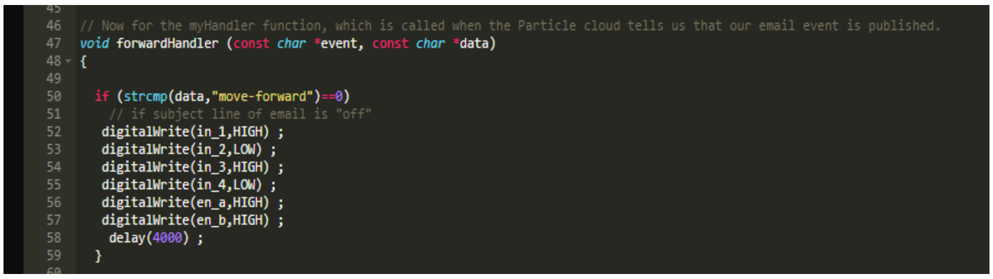
```
void setup()
{

pinMode(in_1,OUTPUT) ;
pinMode(in_2,OUTPUT) ;
pinMode(in_3,OUTPUT) ;
pinMode(in_4,OUTPUT) ;
pinMode(en_a,OUTPUT) ;
pinMode(en_b,OUTPUT) ;
```

Now comes the most important part of the coding. Now we have to code for the subscription of the event. Events were declared while making the applets in the IFTTT server. While making the applets for different functions we have given a proper event name. Like for forward applet the event name was ‘move_forward_event’. so here we will be making the events name and giving them a proper simple name handler so that we can use it further in the codes. We have published the 3 required handlers as per our 3 applets for going front, back and to stop.



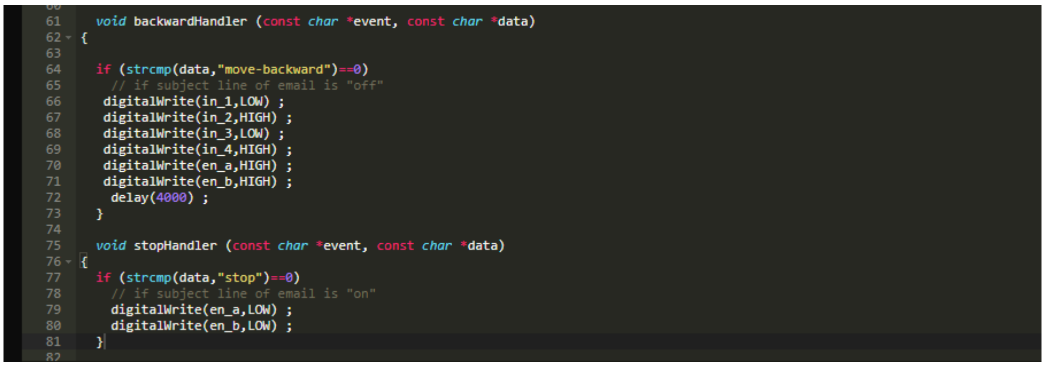
```

 // Subscribe to an event published by IFTTT using Particle.subscribe
    Particle.subscribe("Move_forward_Event", forwardHandler);
    Particle.subscribe("Move_backward_Event", backwardHandler);// Turning on function declaration
    Particle.subscribe("Stop_Event", stopHandler); // Turning off function declaration
  // Subscribe will listen for the event unique_event_name and, when it finds it, will run the function myHandler()
  // (Remember to replace unique_event_name with an event name of your own choosing. Make it somewhat complicated to make sure it's unique.)
  // myHandler() is declared later in this app.
}

```

The next step is to declare the codes for the respective events. Here we are coding for the forward motion of the wheelchair. To do so we have to make both the enable pins of the motor drive HIGH and keep the IN1& IN 3 as HIGH and IN2& IN4 as low. This will make the current to flow into the motor from the 1 and 2 pin of the L298n which will result in the forward rotation of the motors. Here we have also published the event name forwardHandler. This will make sure that when the forward event is been triggered, then the pi should execute only the following codes.



```
// Now for the myHandler function, which is called when the Particle cloud tells us that our email event is published.
void forwardHandler (const char *event, const char *data)
{

  if (strcmp(data,"move-forward")==0)
    // if subject line of email is "off"
   digitalWrite(in_1,HIGH) ;
   digitalWrite(in_2,LOW) ;
   digitalWrite(in_3,HIGH) ;
   digitalWrite(in_4,LOW) ;
   digitalWrite(en_a,HIGH) ;
   digitalWrite(en_b,HIGH) ;
    delay(4000) ;
  }
```

In the same way we have to do publish a backwardHandler for the backward event and the stop event. in back event we will be keeping the IN1 and IN3 as LOW and the IN2 and IN4 as HIGH. This will make the motor to run in reverse direction resulting in the backward motion of the wheelchair. In stop event we will keep both enable pin LOW.



```
  void backwardHandler (const char *event, const char *data)
{

  if (strcmp(data,"move-backward")==0)
    // if subject line of email is "off"
   digitalWrite(in_1,LOW) ;
   digitalWrite(in_2,HIGH) ;
   digitalWrite(in_3,LOW) ;
   digitalWrite(in_4,HIGH) ;
   digitalWrite(en_a,HIGH) ;
   digitalWrite(en_b,HIGH) ;
    delay(4000) ;
  }

  void stopHandler (const char *event, const char *data)
{
  if (strcmp(data,"stop")==0)
    // if subject line of email is "on"
    digitalWrite(en_a,LOW) ;
    digitalWrite(en_b,LOW) ;
  }
```

13. Further applications & future scope

As we can see in this project due to the use of cloud servers like IFTTT and particle server has a lot of advantages. Because of this we can configure our raspberry pi from any part of the world and can flash our codes into it whenever we want without even coming into direct contact with the device. This feature is very handy in many fields like in use of unmaned vehicles in mylitary opperations and in scientific reaserch areas where humans cannot intervene in regular basis. And another advantage of it is its voice module is not connected to the device which is in the case of normal voice controlled wheelchairs. As we have interfaced it with google assistant and we all well know google assistant being a part of any smart device in todays world we can pass on the commands to the device from any location until we speaks in a google integrated device like smart phones.

And as for the future development of this project, its response time can be improved. Like as we can see when the command is given. It first goes to the google server, from there it is been send to the IFTTT server for the trigger action. When it understand the proper trigger it goest to the particle server and unleash the codes under the given subscription. Due to this long channel path the response time of the device gets longer. If we can be abble to eliminate some server out of the channel or can bring some other server which is more properly integrated between each other then the response time can be made short.

References

- http://www.raspberrypi.org, About raspberry Pi
  .
- M. B. Kumaran and A. P. Renold, ”Implementation of voice based wheelchair for differently abled,” 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), Tiruchengode, 2013, pp. 1-6.
- Shafer, M. Turney, F. Ruiz, J. Mabon, V. Paruchuri, Y. SunRobotics based autonomous wheelchair navigation ,J Commun Comput, 13 (2016), pp. 319-328.
- G.S. Pannu, M.D. Ansari, P. Gupta, Design and implementation of autonomous car using Raspberry Pi, Int J Comput Appl, 113 (9) (2015)
- R.C. Simpson, S.P. Levine, Voice control of a power wheelchair, IEEE Transactions on Neural Systems and Rehabilitation Engineering 10 (June (2)) (2002)
- H.G. Nik, G.M. Gutt, N. Peixoto, Voice recognition algorithm for portable assistive devices, IEEE Sensors (2007)
- L. Fehr, W.E. Langbein, S.B. Skaar, Adequacy of power wheelchair control interfaces for persons with severe disabilities: a clinical survey, Journal of Rehabilitation Research and Development 37 (3) (2000)
- A. Kawaguchi, Y. Noda, Y. Sato, Y. Kondo, K. Terashima, A mechatronics vision for smart wheelchairs, in: Proceedings of the International Conference on Assistive Technologies (AT2008), Baltimore, MD, April 2008.
- M. Al-Rousan, WebChair: a web-based wireless navigation wheelchair system for people with motor disability, International Journal of Computers and Applications 27 (5) (2005)
- P. Jia, H. Hu, Head gesture based control of an intillegent wheelchair, in: Proceedings of the 11th Annual Conference of Chinese Automation and Computing Sociecity in UK (CACSUK05), Sheffield, UK, 2005