AetherSync: A Novel Robot to Robot communication device

Hassan Shahzad
Robotics and Artificial Intelligence
University College London
London, UK
hassan.shahzad.23@ucl.ac.uk

Abstract*— We envision a future world with robots going from place to place, building to building picking things up and dropping things off. Some robots may lack levels of intelligence and others more catered to human interaction may have them. However one fundamental thing we forget about so lost in the idea of how humans may interact with robots, is how robots may interact with each other. In this paper I hope to explain how a radio based device could be made which acts as a robots mouth, ears and hands, allowing them to not only interact with each other at short or long distance, but also how they can interact with physical hardware like automatic doors and elevators meaning external costly manipulators on every robot is no longer necessary. This device should build upon existing technology available globally to reduce the amount of funding required to create a platform for which robots can roam freely as well as utilize preexisting popular frameworks such as ROS and custom built libraries for popular programming languages. The communication between robots must also be encrypted in such a way that robots relaying a message must not have access to it nor should humans be able to interfere with the wide area robot mesh network (WARMN)

I. Introduction

We are no doubt in the early stages of the robotic revolution, A time foretold to have intelligent machines walking side by side as humans each capable of interacting with the world as humans do. In place for this to happen, researchers have come up with state of the art navigation algorithms helping robots to go from point A to point B and SLAM algorithms allowing robots to map their environment and know where they are in space. Countless hours of research has gone into how robots can individually traverse the world like humans do yet there is a gap which resides in the idea of that future; the question of how robots interact with each other.

AetherSync is a device that allows the possibility of robot to robot communication regardless of manufacturer or model. A device that can plug directly into any computers central computing system and with the help of well documented and widely used libraries and frameworks, allow robots to speak to each other using a universal language.

Robots within a room can broadcast information to each other to notify surrounding robots of their position in space, data they are reading or even asking for help. The use case for this alone is vast. From swarm robots that self assemble to accomplish tasks that one alone cannot do to robot waiters that could share their point cloud data and generate maps more efficiently and faster by communicating with each other and not being an individual robot.

While there are many use cases of being able to broadcast a message, the robot is limited by how far it can actually send the data. Most low power radio transceivers can only have a range of 300 metres which is also a benefit as it is a downside. A radio module that has too much range means robots have too many incoming messages to parse and handle whereas a small range means it cannot send any useful information to surrounding robots. Hence comes a technology which allows a robot to unicast data within a mesh network. With the use of encrypted packet forwarding, each robot can relay messages given by robot to other robots without access to the data being relayed. This means communication is no longer limited by the range of the radio module but by how many robots there are in the vicinity. If such a network is self organizing and self healing and the method of packet forwarding is such to be the fastest route then a secure means of communication can be established.

A crucial issue which has pushed the development of deployment of robots is that a robot must be able to operate in a world designed for humans. This is the reason why humanoid robots have so much demand in the futire because it is easier to and less expensive to design a solution that can work in a world designed for us than it is to re design our world to be easier for robots. Even a small robot dog must have a costly robot arm attached to it in order to open doors and push buttons. Why can there not be a solution which is in the mid point? A solution where robots do not have to be so expensive to handle the already existing world and one where it is inexpensive to cater the world for robots. AetherSync aims to have another functionality which is intelligent signal emulation. Most automatic doors and elevators seen in office buildings are actually radio controlled operating on a low frequency. Dedicated hand held devices can be used to emulate the required signals to open these automatic doors that most new buildings have. If a robot is able to determine when and how to open a door and which door to open, then with the use of a signal emulating device it should be able to go through doors that were designed for human use without costly manipulators but rather single radio module. This idea is the bridge between both extremes; a world catered for robots and robots with many actuators and careful design to interact with the world around them.

II. Related work

V2V communication is a set of protocols defining how 2 moving vehicles can safely and quickly communicate with one another. While researched over a decade ago, it is only now heavily being deployed in modern electric and hybrid cars with the aim to make roads safer and alert users of what's happening ahead of them.

There are 2 method to enable vehicle to vehicle communication: one is to use DSRC (Dedicated Short Range Communication) and the other is use of cellular (4G and 5G). Different car companies have decide to go down different routes for example Volkswagen has announced only cellular will be used for all of its cars in the future but other companies still choose a short range radio module namely DCRC. One key feature of these protocols associated with the IEEE 802.11p which was modified from IEEE 802.11 to be used in cars is the rapid method of validating a car before communicating with them. Conventional means of validating an unknown agent took to long so a lightweight validation algorithm was employed. This makes sense to use for cars since they move very fast so can quickly leave the range of the DSRC modules however it is not the case for mobiles robots which move car slower. A more rigorous validation algorithm is better to be used for AetherSyncs case.

Zigbee is a wireless protocol that allows for the creation of a self organizing and self healing mesh of nodes where each node is a zigbee radio module. Each node can serve as a coordinator, a router and an end device. The coordinator is the primary controller of the ZigBee network deciding the data which channels through the connection of nodes. The coordinator can communicate with any device in the network including routers and end devices. The router acts as an intermediate node forwarding and receiving messages from other nodes. They help extend the range of coverage of the ZigBee network as each ZigBee devices has a range between 50 to 400 metres depending on the environment . End devices are typically sensors, actuators, or other devices that have specific functions within the network. They are very limited since they cannot participate in routing messages and have a low power consumption profile to save on battery.

The XBee Zigbee module, developed by Digi International, serves as a vital component in various applications of everyday life.It is used extensively in traffic management, outdoor lighting and agricultural use where the distnace between nodes in a mesh network is relatively large and data needs to be routed across long distances.

Each Zigbee module possesses a unique address within the Zigbee network, similar to the MAC addresses found in common consumer electronics. This address ensures the seamless operation of the network by preventing address conflicts. Within a Personal Area Network (PAN), Zigbee devices can engage in unicast communication, allowing nodes to transmit data directly to one another. Additionally, nodes have the capability to forward packets of data, enabling efficient routing within the network. Furthermore, Zigbee devices can broadcast data to directly connected nodes, enhancing the network's flexibility and usability.

This is closely linked to what AetherSync aims to achieve; in this case every robot acts as a node within a large scale robotic mesh network. The difference is that each node should act as a coordinator, router and end device since a robot must send messages, receive messages and securely relay messages.

There are many popular hand held devices that can emulate radio signals to gain access to indoor lights, garage doors, TV's and with some more sophisticated technology, even cars. While technology like this remains controversial in the eyes of legality, it poses as a useful building block for AetherSync. If the use of such a module can be automated and used intelligently with code then robots can open radio enabled doors by themselves.

The Flipper Zero uses the TI CC1101 radio transceiver module for its sub-1 GHz RF communication capabilities. The CC1101 is a low-power, sub-1 GHz RF transceiver designed for very low-power wireless applications. It supports multiple frequencies, including 315 MHz, 433 MHz, 868 MHz, and 915 MHz, making it versatile for various regional wireless communication standards.

III. Prototype photos

A. Initial Prototype

For my initial prototype I wanted to make a minimal working solution for AetherSync which entailed one device broadcasting a string message to another device with no prior information of who the target recipient is. The prototype also allowed a highl evel interface to read and write the messages which was done with the help of a widely used framework called ROS2. The communication was based around the ESP-NOW protocol which Espressif provide built into their microcontrollers. It uses the 2.4GHz band to send information to nearby esp devices. It works in both broadcast and unicast mode within a local mesh network however with the help of further software, the unicast mode can be extended to support packet forwarding allowing a device to send the string further away.

In ROS2, nodes can publish data to a topic and a subscriber can listen to the data that is being sent to the topic. This simple conscept is utillised in many robots where each function of a robot can be broken down into a node with a specific function.

One node might be for handling lidar data and publishing it to the /scan topic and another might be a twist message composed of 3 linear and 3 angular vector values telling a robot how to steer on the /cmd_vel topic. There are many message types that can be sent and received from a topic each designed around specific applications. In my case I went with a simple string message type however later I would like to support all message types to make the idea more scalable.

The inner ROS2 architecture is as follows: a string message that a robot wants to broadcast to the world is published on the /mouth topic. A list of incoming messages from surrounding robots is received from the ear topic by subscribing to it. The same code runs on both microcontrollers that act as nodes in the network however one has an OLED to display the incoming data and data being published.

The bridge between the actual computer running ROS2 and a microcontroller is microROS which is at its simplest level, a library that runs on the microcontroller which can also

subscribe and publish data to the actual ROS2 topic running on the laptop.

microROS is not supported on many microcontrollers but it is on some 32-bit ones; one being the ESP32. The benefit of using an ESP32 is that the same microcontroller can be used to interface with ROS2 aswell as use the ESP-NOW protocol and communicate with nearby devices. This is better than splitting the work across 2 devices or 1 device and a radio module for simplicity and a cheaper solution.

As both microROS and ESP-NOW are running on the ESP32, splitting the time for each process can cause microROS to stop communication to the host computer. The solution is to use freeRTOS and split the communication with the computer and communication to surrounding devices as separate jobs running on the same/different cores. Running it on a different core also has caveats since some fundamental processing for microROS naturally is done on the other core so the priority has to be set very low to prevent interference.

With freeRTOS one job fetches the string to be published and allocates it to an array storing the string. The other job publishes that string with ESP-NOW. The same goes for receiving the data.

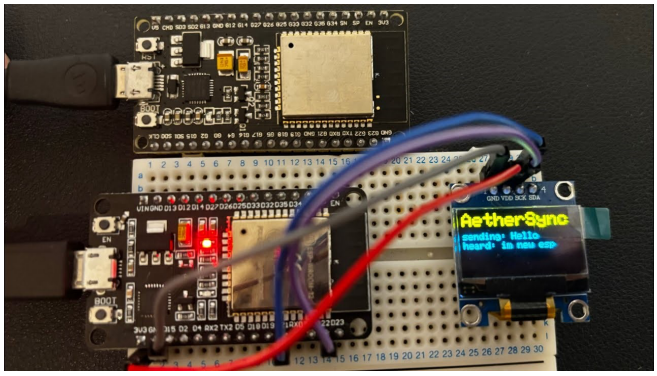


Figure 1: both esp-32 devices are publishing 2 different strings using ROS2. One has an OLED display

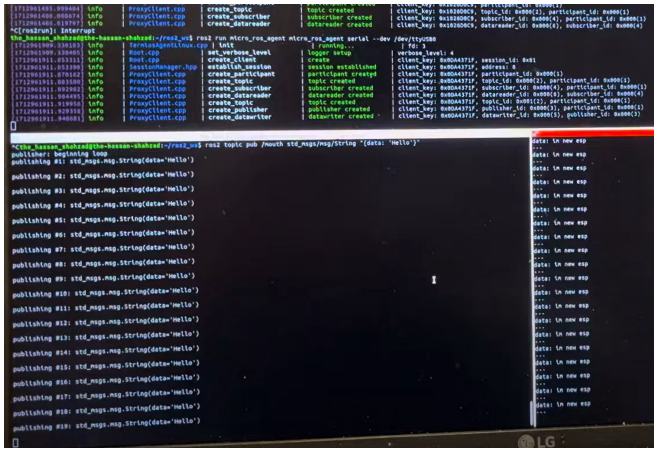


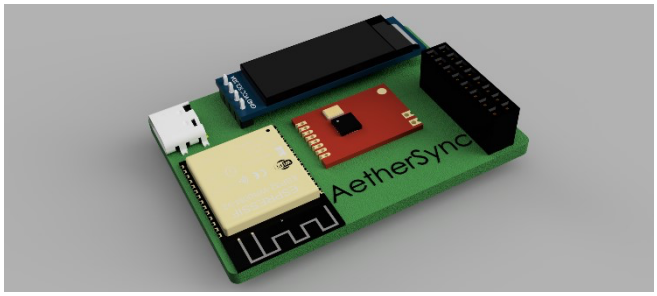
Figure 2: Top terminal shows micro-ROS agent running. Bottom left is ROS2 publishing data to /mouth topic. Bottom right topic is subscribed data from /ear topic

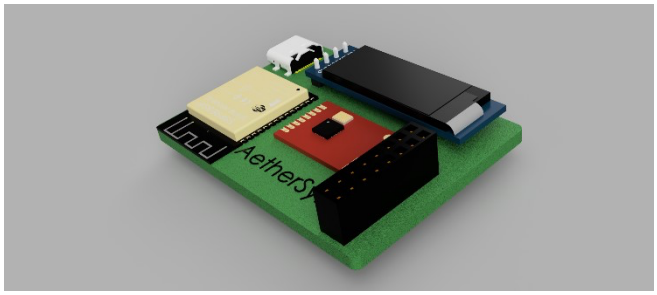
Here we can see each device is capable of broadcasting a string and listening to incoming strings as well as interfacing with ROS2 in a minimal setup. Only 2 topics are used to facilitate a basic broadcasting algorithm

The prototype falls short in a few areas: only broadcasting data is currently supported and you can only send a string message type. The first drawback is an easy fix locally as ESP-NOW has functionality to send data to a specific address however the ability to forward the data to surrounding node in an attempt to reach the target device is difficult and potentially inefficient depending on the method of spreading out the data. Also it would be insecure without a well chosen encryption algorithm since it isn't a good idea for a robot to have access to data that another robot is trying to send through a network.

B. Future prototype

In my next prototype I aim to solve these drawbacks as well as integrate a TI CC1101 chip so that the robot can open automatic doors in its local vicinity and potentially interact with elevators to go from floor to floor inside a building.





Here are some render of what I envision the PCB for AetherSync to look like. It has an OLED to notify the user of information they choose to see such as incoming information within the network or what the robot itself is sending out. It also has GPIO pins making the device more flexible and have use cases in more projects. The pins also have extended use in debugging and interfacing with the host computers in various other protocols like CAN and UART. A Type-c port is used to connect to the central computers USB port for serial communication

AetherSync also has a case to protect it from external damage. The case also has mounting holes to screw onto a robots frame.

IV. Responsible innovation

One of AetherSync fundamental features is the ability to interact with external hardware that have radio chips. This alone reduces the need for a costly, energy intensive robot arm with manipulator. In the long term if we expect a world where robots are common place than the energy consumption would drastically reduce.

If new building begin employing robot friendly architecture where doors, lifts also use AetherSync devices to communicate with surrounding robots then this different in energy consumption would amplify. However for this to happen there needs to be momentum in robots actually being deployed for building stop spend more money in having robot friendly inner architecture.

V. Author bio(s) / experiences

A few years ago I saw a piece of art set in the future where 3 surreal robots where facing each other at a distance and the first thing I wondered is how they spoke to each other. It then raised the question of how will robots speak to each other in the future which led me to work on this device.

My experience making a ROS2 self balancing robot taught me how to use microROS to an advanced level and freeRTOS; both of which are crucially involved in my initial prototype. Here is the link for the project: ROS2 self balancing robot