

WiROS: WiFi sensing toolbox for robotics

William Hunter^{*1}, Aditya Arun^{*2} and Dinesh Bharadia³
^{*}equal contribution

Abstract—Many recent works have explored using WiFi-based sensing to improve SLAM [1], robot manipulation [2] or exploration [3]. Moreover, widespread availability makes WiFi the most advantageous RF signal to leverage. But WiFi sensors lack an accurate, tractable, and versatile toolbox, which hinders their widespread adoption with robot’s sensor stacks.

We develop WiROS to address this immediate need, furnishing many WiFi-related measurements as easy-to-consume ROS topics. Specifically, WiROS is a plug-and-play WiFi sensing toolbox providing access to coarse-grained WiFi signal strength (RSSI), fine-grained WiFi channel state information (CSI), and other MAC-layer information (device address, packet id’s or frequency-channel information). Additionally, WiROS open-sources state-of-art algorithms to calibrate and process WiFi measurements to furnish accurate bearing information for received WiFi signals. The open-sourced repository is: [WiROS:https://github.com/ucsdwcsng/WiROS](https://github.com/ucsdwcsng/WiROS).

I. INTRODUCTION

Incorporating wireless sensing into a robotics sensor stack allows robots to better handle the failure cases of visual sensors like cameras or Lidars. Many recent works [1], [2], [4] have shown that integrating wireless signals helps to overcome common failure cases like visual occlusion, dynamic lighting, or perceptual aliasing [5]. Among the various wireless-sensing modalities explored, leveraging WiFi sensing is particularly attractive. It is widely deployed in many indoor environments, most robots already have WiFi chipsets for communication purposes, and it provides an extended sensing range (over 10 m). With this in mind, many recent papers [1], [4], [6] explore using WiFi signals.

However, no easy solutions exist for integrating them into an existing robot’s sensor stack, precluding their wider use. Popular cameras [7] and Lidars [8] enjoy wide support within the Robot Operating System (ROS) framework with clear documentation. This ROS support allows for quick integration and easy operation of these sensors. The same, however, cannot be said for WiFi sensors, inhibiting their adoption for robotics applications. In this work, we present WiROS to fill in this gap. WiROS develops a set of ROS nodes that furnishes raw WiFi measurements, processed WiFi features, and visualizations as ROS topics and provides an easy-to-perform wireless calibration framework.

With the broad goal of providing accessible WiFi sensing

¹William Hunter is with Electrical and Computer Engineering Dept., University of California, San Diego, CA 92092, USA wshunter@eng.ucsd.edu

²Aditya Arun is with Electrical and Computer Engineering Dept., University of California, San Diego, CA 92092, USA aarun@ucsd.edu

³Dinesh Bharadia is faculty with Electrical and Computer Engineering Dept., University of California, San Diego, CA 92092, USA dineshb@eng.ucsd.edu

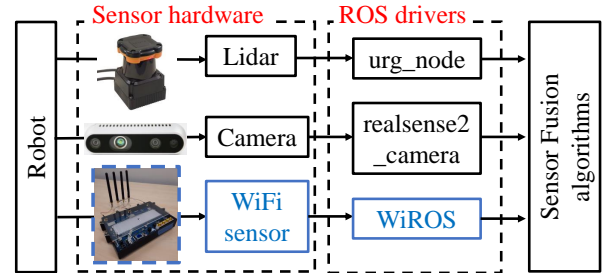


Fig. 1: The sensor hardware, like cameras and Lidars, have widely supported ROS nodes. This work releases WiROS, a similar ROS node compatible with WiFi sensors.

information within robot sensor stacks, we follow three design principles. A WiFi sensor wrapper should be

- 1) **Accurate**: provide WiFi sensing measurements accurately,
- 2) **Tractable**: quick to bring up and easy to calibrate,
- 3) **Versatile**: widely usable in WiFi-equipped spaces.

Prior Works: However, many existing systems that furnish WiFi measurements at the robot application layer fail to meet one or a few of these requirements, making them unsuitable for robot integration.

- **WiFi measurement tools:** There are two widely used open-source WiFi measurement toolkits [9], [10] which support the IEEE 802.11n protocol. However, this protocol is outdated and consequently not as widely supported in current WiFi deployments. Alternatively, newer toolkits that leverage the IEEE 802.11ac [11] and IEEE 802.11ax [12], [13] exist. Unfortunately, these systems do not expose WiFi measurements in real-time, instead storing it on the device in specialized files. This requires additional post-processing and time-synchronization, precluding real-time robot operation. Consequently, these toolkits are neither versatile nor tractable.
- **ROS-supported Tools:** To circumvent these challenges, a few tools integrate WiFi measurements into ROS frameworks. [14] provides only the WiFi signal strength (RSSI) measurements via a ROS topic, which can be used as a proxy between a transmitter and receiver. However, RSSI is a very coarse-grained, environment-sensitive measurement, whereas richer WiFi measurements exist that can expand the use of WiFi sensors. These richer measurements (channel state information, CSI) are exposed in the tools above and help determine the arrival and departure angles of a WiFi signal, the velocity of the WiFi sensor, or even fine-grained information about the environment [15]. Figure 2 details these physical measurements. A recent work [16] looked at providing ROS support for the 802.11n chipsets [9], [10] mentioned previously. However, this

system requires active collaboration with the WiFi infrastructure, requiring deployed WiFi APs to perform ‘round-robin’ packet exchanges. This infrastructure dependency precludes ubiquitous deployment of WiFi sensing in indoor environments and presents logistic, security, and networking challenges. For instance, a public deployment of their system would require firmware upgrades on deployed WiFi access points, authentication of third-party robots with a secure network, and introduce additional burdens on a congested network. Additionally, as of this writing¹, it does not provide ready integration to ROS.

WiROS instead leverages widely used 802.11ac WiFi protocol, which provides $2\times$ the sensing bandwidth (improving *accuracy*). We build on prior work [11] a WiFi sensing framework making the following contributions and overcoming the limitation of the prior works:

1. Scalable framework for WiFi sensing: We provide a ROS node (‘CSI-Node’) as a simple abstraction over the hardware used to collect CSI to allow for simple integration and leverage ROS-based time synchronization. WiROS subsequently extracts various physical parameters (angles of arrival and departure of the signal, angular positions of reflections in the environment) to aid robot operation. For example, WiROS can measure the bearings of other robots or arbitrary transmitters (WiFi APs, laptops, phones, or IoT devices) from a single received WiFi packet, without associating or authenticating with the existing network. This prevents network congestion, does not require collaboration with network infrastructure, and allows for ubiquitous deployment of our system in multiple indoor spaces. Clearly, WiROS enables *versatile* WiFi sensing, and additional details can be found in Section II-B.

2. Easy calibration and setup: Commercial APs come with unique hardware biases, which can skew the signal measurements [17]. These biases need to be measured a priori, and measurements must be calibrated to estimate various physical parameters, including the bearing of the transmitted signal. Past works [18], [19], [16] have required disassembly of the device and manual measurement of these biases, severely reducing the tractability of a sensor platform. Instead, we provide a solution for automatically calibrating the phase offsets on-robot by extending the work in [17]. This allows for hassle-free and real-time wireless calibration, with Section II-C providing further details.

3. Algorithms for sensing: Finally, we provide a ROS node (‘Feature-extraction Node’) to estimate the angles of arrival and departure of a WiFi signal using state-of-the-art techniques [20], [21]. Additionally, we provide intuitive visualizations of the received WiFi signal to aid debugging. Providing this node serves two key purposes. First, it allows out-of-the-box use of WiFi measurements for SLAM and other navigation purposes. Second, it provides a blueprint for using WiFi CSI to perform wider tasks like Doppler estimation or time-of-flight measurements [22]. These concepts are further elaborated in Section II-D.

In the next section, we will provide a high-level overview

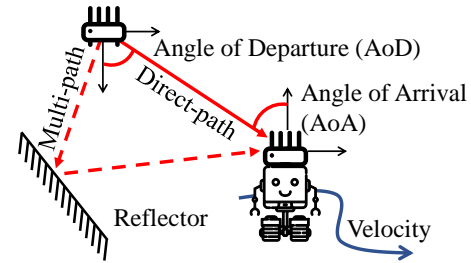


Fig. 2: Signal parameters measurable by CSI The direct path’s (solid line) and multi-path’s (dotted line) angles of arrival and departure can be measured in the local coordinates of the APs. The robot’s velocity may also be measured.

of WiROS’s usage (Section II-A) and specific details about the above three contributions.

II. DESIGN AND USAGE

WiROS’s primary motivation is to provide an accessible WiFi sensor within robot sensor stacks to leverage key advantages WiFi signals provide. In this vein, we will first provide a quick rundown on the usage of WiROS, the WiFi measurements exposed via ROS topics and visualizations of these measurements (Section II-A). This will act as an overview for the extensive documentation provided along with the code repositories²

WiROS’s secondary motivation is to provide a replicable framework that can be easily integrated with different WiFi radio hardware and CSI extraction toolkits. In this vein, we will provide details of

- (a) extending Nexmon-CSI [11] as an example in developing WiROS (Section II-B),
- (b) the wireless calibration techniques incorporated within WiROS (Section II-C), and
- (c) the open-sourced implementation of various state-of-art bearing estimation algorithms for readily using WiFi measurements (Section II-D).

A. Using WiROS

First, we demonstrate usage of WiROS with Asus RT-AC86U [23], however, other chipsets can also be readily used, and we provide further details on how one can integrate other chipsets into WiROS’s framework in Section II-E. To setup the hardware, we need to follow three simple steps

- 1) Compile the open-source ROS packages using `catkin`.²
- 2) Setup the access point by flashing the provided firmware.
- 3) Configure provided ROS `params` and stream the WiFi measurements as ROS `topics`.

To elaborate on these steps, we will set up this Asus off-the-shelf access point as both a WiFi receiver or a transmitter using WiROS’s CSI Node². As shown in Figure 3, the CSI Node takes as input ROS `params`, in blue, which will be discussed briefly below.

Setting up a receiver: The specific WiFi frequency-channel and bandwidth should be first configured via ‘channel’ `params`. A filter for specific MAC addresses can also be included via

² WiROS’s Index page: <https://github.com/ucsdwcsng/WiROS>
 Sub-links: CSI Node: https://github.com/ucsdwcsng/wiros_csi_node
 Feature-extraction Node: https://github.com/ucsdwcsng/wiros_processing_node
 Custom RF messages: https://github.com/ucsdwcsng/rf_msgs

¹Referring to Github commit 7e79508

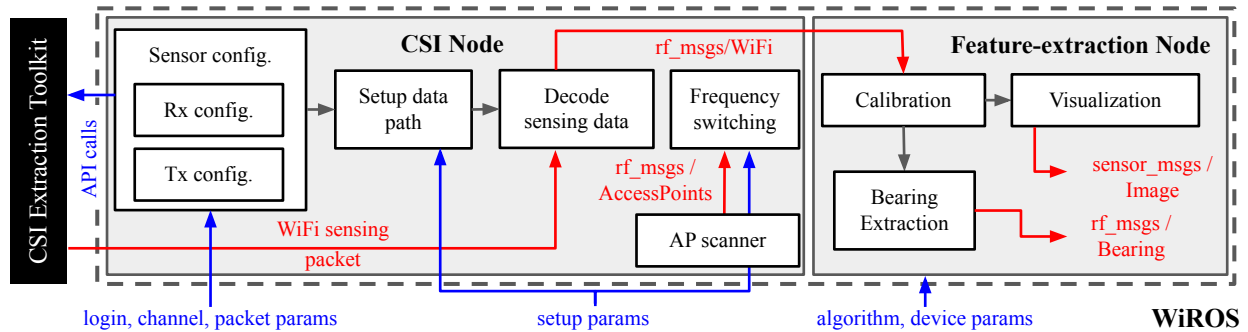


Fig. 3: WiROS’s Block Diagram: An inside look at the WiROS’s box from Fig 1. Showcases the two main blocks - CSI Node and Feature-extraction Node to extract raw WiFi measurements and to calibrate and process these measurements. The blue text indicates the control plane parameters, whereas the red text indicates the exposed measurements. WiROS extends the functionality of the underlying black box ‘CSI Extraction Toolkit.’

the ‘packet’ params. If an all-pass filter is used, WiROS will monitor and provide WiFi measurements for received signals for all WiFi devices transmitting on the specified WiFi channel and bandwidth. With these configurations set up, the receiver can be brought up with `roslaunch wiros_csi_node csi_node`. Alternatively, simple `launch` files are provided to further improve *tractability*.

However, as mentioned, receiving WiFi signals from a specific channel and bandwidth can reduce usability in a wild environment. Most enterprise WiFi networks deploy multiple access points (AP) that serve users across a building. Neighboring APs in these networks are often deployed on different frequency channels to reduce interference. Hence, WiROS will miss out on CSI measurements from APs configured on other channels. To ensure *versatility* of our system, we implement a ROS `service` which allows a user to switch channels on the WiFi sensors. Additionally, WiROS automates this channel switching to monitor the channel of the nearest AP by periodically scanning all the WiFi channels. This automated behavior can be enabled via the ‘setup’ params.

Setting up a transmitter: Additionally, researchers may often require a steady source of WiFi transmissions, with configurations of the WiFi frequency-channel, bandwidth or MAC address. For example, later in Section II-C, we will need to configure a WiFi AP to transmit packets continuously to compute the hardware calibration parameters. The ‘channel’ and ‘setup’ params can be used to configure the transmitter. With these parameters configured, the AP can be brought up with the same `roslaunch` command as before. Next, we will discuss an overview of the exposed measurements available as ROS `topics` via an Ethernet connection.

Exposed measurements: Once a receiver has been set up, it will capture WiFi measurements from ambient WiFi signals in the configured frequency-channel and bandwidth. These measurements will be available via a ROS `topic` through a custom WiFi-sensing ROS `message`, `rf_msgs/Wifi` as shown in Figure 3. Amongst the various exposed information, the signal strength (RSSI) and the channel state information (CSI) are the two commonly used measurements. RSSI, measured in dB, provides a rough estimate of the distance of the transmitter and is widely used as a proxy to measure

communication throughput [24], as a feature for improving SLAM performance [6] or to map the transmitters [25] coarsely. Alternatively, CSI provides more fine-grained information. It is a matrix of complex numbers indicating the received signal magnitude and phase across a set of transmitted frequencies and receiver antennas. The number of transmitted frequencies is controlled by the bandwidth (20, 40 or 80 Mhz), with the 80 MHz bandwidth only supported in the 802.11ac protocol. By default, for the Asus hardware, there are 4 receive antennas compared to 3 receive antennas commonly present in alternative systems [10], [9]. The larger bandwidth and number of antennas provide more *accurate* sensing capabilities. Additionally, if a transmitter has multiple antennas configured, an additional dimension for the number of transmit antennas are present. However, these CSI measurements are challenging to visualize. To improve the *tractability* of WiFi sensors, we provide further details on the real-time visualizations included in WiROS next.

Visualization and processing measurements: The CSI measurements available via ROS `topics` can be processed and visualized by deploying the Feature-extraction node². This node furnishes two important visualizations – the magnitude-phase profile and the bearing-range likelihood profile. The magnitude-phase profile (see Figure 4(a)) provides a visualization of the CSI measurements, providing the magnitude (y-axis of top image) and phase (y-axis of bottom image) of the received signal in decibels and degrees, respectively, across various subcarrier frequencies (in the x-axis) and across the four receive antennas (as different lines). The bearing-range profile (see Figure 4(c)) indicates the arrival of a WiFi signal from the transmitter via a straight-line direct path and numerous reflected paths. The profile’s peaks indicate the bearing of the transmitter (in the vertical axis) and the relative distances of the direct-path signal and its reflected-path echoes. Additionally, visualization of these profiles is provided in the supplementary video.³ Both these visualizations, exported as `sensor_msgs/Image` as shown in Figure 3, can be utilized to debug the wireless channel, check if all the receive antennas are performing equivalently or tune processing parameters like algorithms to leverage, time-window for averaging or RSSI thresholds. Additionally,

³ Youtube demo link: <https://youtu.be/zYAshWF75k>

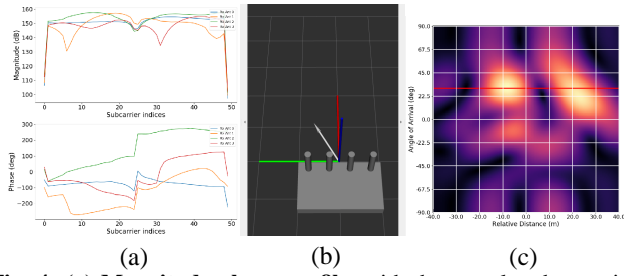


Fig. 4: (a) Magnitude-phase profile, with the top plot showcasing the magnitude and bottom plot the phases across 4 receivers and 234 subcarrier frequencies. **(b)** A simplified visualization of the predicted bearing of the signal (white arrow) in the local coordinates of the access point. **(c) Bearing-range likelihood profile**, with the red-line indicating the strongest received path, and hence the bearing of the signal. These images are visualized via the ROS RViz software in realtime

a simple visualization of the received signal’s bearing (white arrow) in the access point’s local coordinates is shown in Figure 4(b).

Additionally, the Feature-extraction node open-sources state-of-art bearing estimation algorithms [20], [1], [21] to measure the angles of arrival and departure of the received signal. These measurements are processed in real-time and exposed as ROS topics via custom WiFi-bearing messages, `rf_msgs/Bearing`.

This brief description showcases the ease of use of WiROS’s WiFi sensor for real-time data collection, processing, and debugging, improving state-of-the-art systems that require cumbersome data file post-processing. In addition to this description, extensive setup and usage documentation is provided with the code-repositories². The next section will provide further specifics about the implementation details and discuss the specific process block in Fig. 3.

B. CSI Node: Middleware to integrate with ROS

The ‘CSI node’ is modularly written in C++, allowing it to interface with multiple CSI extraction toolkits (the black box in Figure 3). It is currently tested with Nexmon-CSI’s [11] Asus RT-AC86U [23] platform as it is the most capable extant solution for COTS CSI extraction. The CSI Node is tested on ROS Kinetic, Melodic, and Noetic. It runs out-of-the-box on a Raspberry PI, or via a dedicated Ethernet connection to a central server.

Data Path: WiROS takes as input raw CSI data packets from the black box (shown in red arrows) via a UDP socket configured over an Ethernet connection by the ‘Setup Data-path’ block. The ‘Decode Data’ block decodes these raw WiFi data packets to expose the measurements in easy-to-consume `rf_msgs/WiFi` format.

Control Path: The CSI-node additionally provides different radio configurations via API calls. This abstracts out the specific hardware implementation for the end user. A dedicated ‘Sensor Config’ submodule configures the underlying WiFi radio. It requires ‘login’ params to access the WiFi AP, ‘channel’ params to configure the WiFi frequency-channel and bandwidth, and ‘packet’ params to configure the transmitter’s beacon rate or filter WiFi data packets.

Functioning in parallel, two sub-modules, if enabled via

the ‘setup’ params, are responsible for tracking and adjusting the frequency-channel used by the AP. The ‘AP scanner’ periodically scans the different channels and determines the closest WiFi AP to re-configure the WiFi radio by exposing the information via the `rf_msgs/AccessPoints` message. The ‘Channel switching’ sub-module handles switching the channel with minimal sensing downtime (less than 500 milliseconds).

C. Quick and easy calibration

Calibrating a sensor is a necessary first step and must be easy to perform. Generally, the calibration can vary for different frequency-channels and is unique for each hardware. This necessitates an easy-to-deploy and accurate calibration framework for the tractability of the WiFi sensor. This section will elaborate on the one-time wireless calibration system provided via a Python3 script in the ‘Feature-extraction’ node.

Usage: To calibrate our raw WiFi messages collected in the previous section, we must apply independent phase corrections across each antenna and frequency measurement. To compute this calibration, first, configure a receiver and transmitter to a specific WiFi frequency-channel as discussed in Section II-A. Next, collect raw CSI measurements by placing a WiFi sensor ‘receiver’ on a robot and a WiFi sensor ‘transmitter’ in a static predefined location in space. Instead of an ASUS WiFi sensor, a phone or laptop may be configured to transmit ‘ping’ packets. In this setup, we are looking to calibrate the WiFi sensor receiver. Run the robot in any pattern in relatively free space, within a 5 m radius of the transmitter, and collect the robot odometry measurements (\vec{r}_t) as `nav_msgs/Odometry` and the WiFi measurements (W_t) as `rf_msgs/Wifi` from the CSI node. Note the location of the transmitter (\vec{t}) in the robot’s generated map (often visible when using a LiDAR) and measure the relative antenna locations (\vec{a}_i) on the receiver. This data can be input into the calibration framework to generate the wireless calibration matrix, provided with the ‘Feature-extraction’ Node².

Behind the scenes: From our discussion so far, we need to find a phase calibration matrix C ,

$$C = \exp(j\Phi) \in \mathbb{C}^{4 \times N_f},$$

to calibrate the phase measurements across the 4 antennas and N_f frequencies, where $\Phi \in \mathbb{R}^{4 \times N_f}$. Given a raw CSI measurement from the CSI Node, $W_t \in \mathbb{C}^{4 \times N_f}$, the calibration is applied as

$$W_t^{\text{cal}} = C \odot W_t,$$

where \odot is the Hadamard (element-wise) product.

Using the robot poses ($\vec{r}_t = (r_t^x, r_t^y, r_t^\theta) \in SE(2)$) and transmitter location ($\vec{t} \in \mathbb{R}^2$), we first compute the expected ground truth bearings (θ_t). These can then be converted to expected WiFi CSI measurements ($\hat{W}_t \in \mathbb{C}^{4 \times N_f}$, implicitly assuming 4 receive antennas).

$$\theta_t = \frac{\pi}{2} - \left(\arctan \left(\frac{r_t^y - t^y}{r_t^x - t^x} \right) - r_t^\theta \right)$$

$$\hat{W}_t^{i,j} = \exp \left(\frac{2\pi j}{\lambda} [\cos(\theta_t), \sin(\theta_t)] \vec{a}_i \right) \quad \forall j \in [1, N_f], \forall i \in [1, 4]$$

where, \vec{a}_i is the relative location of the i^{th} antenna with respect to the first antenna; consequently, $\vec{a}_1 = \vec{0}$. λ is the wavelength of the center frequency for the WiFi channel in consideration. Assuming a strong line-of-sight path signal is present in our measurements, we can expect the phases $\angle \hat{W}_t \approx \angle W_t^{\text{cal}}$. Note that our assumption is reasonable given that the calibration data is collected in a relatively open environment with no blockages to the signal. Consequently, we can suppress the phase difference induced by bearings as

$$W_t^{\text{sup}} = W_t \odot \text{conj}(\hat{W}_t)$$

This leaves the remaining calibration phase C in W_t^{sup} . However, each WiFi measurement may have multiple reflected paths and hardware-centric Gaussian noise, which have not been adequately suppressed. However, we have two hints. One, reflections are inconsistent across different locations; two, averaging can suppress Gaussian noise. Hence, the best calibration estimate is the strongest remaining component in the suppressed W_t^{sup} measurements. We can leverage Principle Component Analysis to extract this strongest component in our calibration data as

$$\begin{aligned} W_t^{\text{flat}} &= \text{flatten}(W_t^{\text{sup}}), \quad W_t^{\text{flat}} \in \mathbb{C}^{4N_f} \\ \mathbb{W} &= [W_0^{\text{flat}} \quad W_1^{\text{flat}} \quad \dots \quad W_T^{\text{flat}}] \\ \mathbb{U}, \mathbb{S}, \mathbb{V} &= \text{SVD}(\mathbb{W}) \\ \Phi^{\text{coarse}} &= \angle \text{reshape}(\mathbb{U}_0), \quad C^{\text{coarse}} = \exp(j\Phi^{\text{coarse}}) \end{aligned}$$

where ‘flatten’ converts the matrix into a vector, SVD computes the full singular-value decomposition, ‘reshape’ converts the vector back into a matrix of the original dimensions, and \angle computes the phase of the complex numbers. \mathbb{U}_0 is the first and strongest principal component of \mathbb{W} . However, as indicated, this calibration is a coarse estimate. The expectation is for the calibration matrix to consist of unit-norm *elements*, but the principle component, \mathbb{U}_0 , is a unit-norm *vector*, violating this property. Hence, we need to re-project C^{coarse} onto a valid space of calibrations. To find a valid calibration, C^{fine} , that is close to C^{coarse} , we note that C^{fine} must be orthogonal to the other vectors in \mathbb{U} , so we try to find a fine-tuned calibration Φ^{fine} which has the lowest norm when it is projected onto $\mathbb{U}_{[1:]}$.

$$\begin{aligned} \Phi^{\text{fine}} &= \min_{\Phi} \|\mathbb{U}_{[1:]}^T \text{flatten}(\exp(j\Phi))\|_2^2 \\ C^{\text{fine}} &= \exp(1j\Phi^{\text{fine}}) \end{aligned}$$

where $\mathbb{U}_{[1:]}$ is the orthogonal space, and we minimize for Φ by leveraging the Levenberg-Marquardt [26] algorithm with an initialization of Φ^{coarse} . Via this fine-tuning process, we recover the wireless phase calibration matrix C^{fine} . Later in Section III, we will evaluate the accuracy and versatility of this calibration across different WiFi frequency channels and hardware restarts.

D. Feature-extraction Node: Library of CSI processing tools

The measurements from the ‘CSI Node’ can be leveraged to measure signal path parameters like signal strength, angles of arrival or departure, the velocity of the transmitter, or locations of reflections in the environment. However, angular

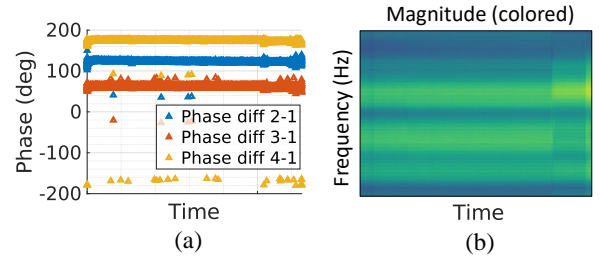


Fig. 5: Stability of WiFi measurements: WiFi measurements collected over 18 hours showcases (a) **phase stability:** CSI phase at DC frequency for 4 receiver antennas and, (b) **magnitude stability:** CSI spectrogram across all transmitted frequencies for an 80 MHz bandwidth signal.

information can be readily measured from a single packet and provides a quick way to realize the advantages of WiFi measurements. In this vein, we open-source a large set of state-of-art bearing estimation tools [20], [21] and CSI filtering techniques.

Usage: Deploy WiROS’s receiver on the robot and compensate the hardware as previously mentioned. We can set up WiROS’s transmitter or monitor existing WiFi packets in the environment to measure the CSI measurements using the CSI-Node. Leveraging these CSI measurements, we can compute the angle of arrival (AoA) of a signal at the receiver (robot-side bearing) and the angle of departure of a signal at the transmitter (AP-side bearing) from a single WiFi signal received at the robot. These bearings would be in the robot’s and the AP’s local coordinate frames and are exposed via the `rf_msgs/Bearing` ROS *topic*. The ‘Feature-extraction’ node also generates specialized visualizations to help debug the wireless channel, as discussed in Section II-A.

Behind the scenes: The ‘Feature-extraction’ node is written in Python3, as this allows easy modification by users to support their specific sensing needs. It collects and sorts CSI messages and passes them to a data consumer object, collating the CSI measurements from a specific target device and using them to generate bearing estimates. We provide several example consumers which run a variety of AoA algorithms, from a very compute-efficient bearing-only algorithm that can be run in real-time to computationally heavy algorithms like SpotFi[20]. Additionally, we note that CSI measurements can often fluctuate during motion because of dynamic reflections in the environment. In situations where we need to measure the direct signal path, these fluctuations can be detrimental. Hence, to obtain reflections-suppressed CSI measurements, we also implement an averaging technique from our prior work [1]. Supporting math for these algorithms is provided with the documentation².

E. Extending WiROS to other CSI toolkits

The CSI Node and Feature-extraction Node form the cornerstones of WiROS. The CSI Node interfaces WiROS with the underlying hardware, whereas the Feature-extraction Node open-source state-of-the-art post-processing algorithms for the WiFi CSI measurements. WiROS currently extends the Nexmon CSI extraction toolkit [18], making specific API calls to configure the underlying hardware. However, we can follow a similar design template to adapt WiROS to other CSI

toolkits [12], [13] as well. Additionally, we encourage future hardware platforms and CSI-extraction toolkits to prioritize exposing data via a UDP socket. Consequently, these newer platforms can enjoy ROS support with minimal changes to WiROS’s ‘Sensor Config’ and ‘Decode data’ modules.

III. VERIFICATION STEPS

In the following section, we will use the Asus AC86U [23] (henceforth just WiFi sensor) as an example to verify the various aspects of WiROS. We deploy this WiFi sensor on a Turtlebot 2 platform [27] equipped with a Hokuyo Lidar [8] and Realsense Camera [7]. We collect all data on a Thinkpad 13” Laptop [28]. Specifically, we care about the stability of our signal measurements, the efficacy of our wireless calibration techniques, and the performance of the bearing estimation and automated channel switching feature. These sets of experiments provide a guideline for testing a WiFi sensor. If the specified Asus hardware is used, the reader should expect similar results as showcased. However, WiROS can be easily extended to other hardware systems as well, in which case, the following sets of verification can be performed for the new sensor.

A. Measurement stability

First, we ensure our sensor provides stable measurements over time, unaffected by the board’s temperature differences and minor disturbances to the setup (as may be expected when the sensor is deployed on the robot). Two WiFi sensors were placed in an environment. One was set up to transmit data at 1 Hz and another to receive this transmitted data (see Section II-A), which were subsequently stored in a *bag* file.

In Figure 5(a), we showcase the phase stability across antennas at the center frequency over an 18 hour run. We plot the same for the magnitude of the received signal (in color intensity) across the different frequencies (in the y-axis) for a single antenna over time (in the x-axis) in Figure 5(b). These plots show little variation in the phase and magnitude measurements over time, indicating that the CSI measurements are reliable and consistent.

B. Calibration efficacy

Next, we must effectively correct hardware biases and offsets to measure accurate bearings. Specifically, we observe that without appropriate calibration, our median bearing errors can be 115° . However, post calibration, we can expect median bearing errors of 5.3° . Hence we provide a wireless calibration system as described in Section II-C as a core part of WiROS. However, this technique must work consistently for different WiFi channels and upon device reset.

We collect the data as explained in Section II-C to confirm the calibration performance. We place an additional sensor in the environment sending beacon packets and another on the robot receiving them. We run the robot in a random pattern, collecting CSI data within *bag* files, as shown in Figure 7(a). This is additionally done on different WiFi channels. Next, we re-run these experiments in different environments (shown in Figure 6) to test the calibration efficacy.

Figure 7(b) shows bearing errors across different channels after the APs were reset post collecting the required training

data. For this experiment, we measure calibration for different channels and test the quality of these calibrations by collecting a test dataset after restarting the AP by characterizing the bearing accuracy. We observe similar performance across all the 80 MhZ WiFi channels [29]. Additionally, note that these calibration properties are specific to the Asus hardware used in this verification process. The wireless calibration procedure provided is agnostic to hardware, but it is imperative to take these verification steps when working with different hardware.

C. *Bearing Estimation accuracy of open-sourced algorithms*
 Post calibration, we have corrected phase measurements which can be used for bearing estimation. In the previous section, we presented some bearing results to showcase calibration efficacy. However, we glossed over the specific algorithms used. As discussed in Section II-D, we open source implementation of prior bearing estimation techniques [20], [21]. More details of these algorithms can be found in our documentation of the Feature-extraction node². The accuracy of these algorithms is shown in Figure 8(a). In these experiments, we ensure to include only received packets that have received signal strength stronger than -65 dB. This allows us to reject outliers. However, it is important to note that this threshold is a hardware-specific number, and other hardware systems may require different thresholds.

D. Feature: Automated channel switching

We test the channel switching feature as explained in Section II-A in an enterprise network deployed in our university building in Figure 8(b). The robot traverses twice around the environment, shown in overlapping trajectory, in a single run. The colors represent the different APs deployed as part of the enterprise network, and the colors along the path indicate the specific AP WiROS’s sensing channel is tuned into. The figure showcases that at different robot locations, WiROS appropriately tunes to the nearest AP, ensuring that the robot can use WiFi-sensing packets from line-of-sight access points. Furthermore, channel switching also depends on the path taken in the environment, which is apparent from the different behaviors across the two traversals. WiROS prioritizes continued interaction with the current AP rather than frequent AP switching to reduce downtime and consistency of data collection.

IV. CASE-STUDIES

Next, we present three case studies to showcase the applicability of WiROS’s various features. The corresponding Python3 scripts to get started with WiROS and these case studies are provided within the ‘Feature-extraction’ node.

A. Localization to combat Kidnapped Robot Problem

In the kidnapped robot problem, a robot carried to an arbitrary location in the environment is lost as it fails to re-localize within the global map. This is an additional challenge in GPS-denied scenarios where a robot does not have a global location estimate, which is common in indoor scenarios. However, some works [31] have used WiFi estimates to tackle this problem by providing global location estimates.

WiROS can be leveraged to solve the kidnapped robot problem in indoor settings by leveraging existing deployed

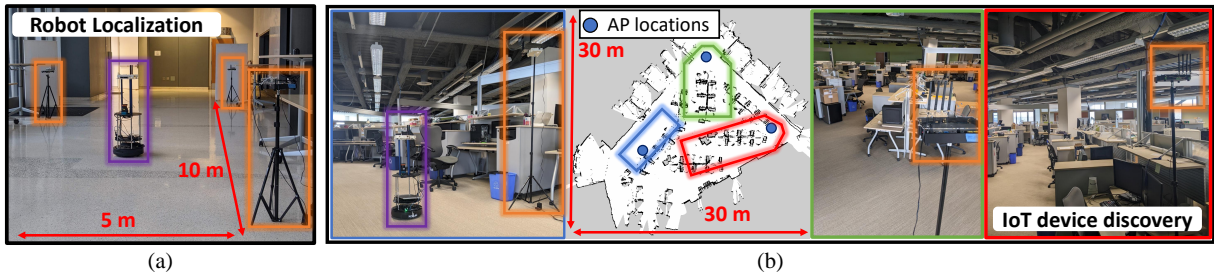


Fig. 6: Testing Environments for case-studies (a) Kidnapped robot problem and (b) IoT Localization. The purple and orange boxes show the robot (with WiROS’s receiver) and WiROS’s transmitter. In (b), the colored photo frames (blue, green and red) correspond to the camera viewpoint in the 2D top-down map.

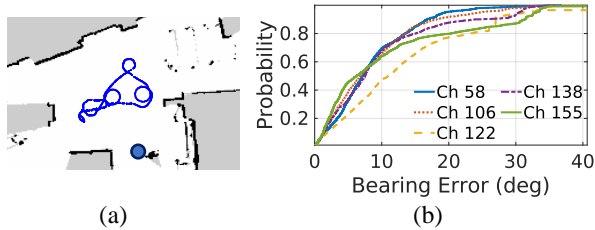


Fig. 7: Consistency and accuracy of calibration (a) Robot path and transmitter location (blue circle) to calibrate the robot’s WiFi sensor in 3×3 m, (b) Bearing errors when calibrated across different channels, after reset and channel switching

WiFi infrastructure as shown in Figure 8(c, i). By configuring WiROS’s receiver to sense all WiFi packets in the environment, we can collect WiFi measurements from all nearby APs. This is made easier by the provided automated channel-switching feature. Next, WiROS can estimate the robot’s bearing in the AP’s frame of reference by leveraging the angle of departure (red arrows). These AP-sided bearings can be subsequently used to triangulate the robot. We need prior knowledge of the AP’s location, antenna array geometry, and calibration metrics (which can be wirelessly computed using the ‘Feature-extraction’ node) to deploy this application. As a test, we deploy 4 access points in a 5×10 m environment (Figure 6(a)) and observe a median localization accuracy of 1.2 m.

B. Odometry-correction in SLAM systems

Erroneous sensor measurements made by odometry sensors, cameras, or Lidars can create drift in a robot’s predicted location. These drifts are corrected by applying “loop closures”, where a robot leverages the intuition that it must predict similar location estimates when revisiting previously seen spaces. However, due to perceptual aliasing [5], different locations in the map may look similar, creating ambiguity when detecting loop closures. However, recent works [1], [6] have looked at fusing WiFi-bearing measurements to address the problems with visual loop closures.

Again, WiROS can be leveraged to solve the issues with sensor drift during loop-closure as shown in Figure 8(c, ii). We can configure WiROS’s receiver at the robot to collect WiFi measurements from APs in the wild. Subsequently, the measured robot-sided and AP-sided bearings from these WiFi measurements (red and blue arrows) can be fused with robot odometry in a factor graph to correct for drifts. Sensor fusion is made easy by WiROS as WiFi-based bearing measurements are readily available as ROS topics and time-synchronized with other sensor measurements on the robot.

In Fig. 8(d), we compare WiFi-based bearings fused with odometry from Visual-Inertial odometry (WiFi+VIO, no loop-closures) and a state-of-the-art Visual-inertial SLAM system, with loop closures enabled [30] (VIO only). We observe a 30% reduction in median error due to improved robustness to visual aliasing.

C. IoT device mapping to aid device management

Finally, instead of localizing a robot in an environment (which is carrying a WiFi transceiver), we can consider the reverse problem of localizing WiFi transceivers in an environment. This is an important problem for IoT device management [32], where localizing various devices in a large space within a common map is necessary to aid building managers in maintaining the different IoT devices. Alternatively, recent works [25] have looked at localizing, potentially rogue, IoT devices to ensure the privacy and security of users.

WiROS can be easily leveraged to localize all WiFi IoT devices in a given space, as shown in Figure 8(d, iii). A robot deployed with WiROS’s receiver, scanning all channels and enabling an all-pass MAC filter, can capture WiFi measurements from all WiFi-based IoT devices in space. Subsequently, the robot-sided bearing measurements (blue arrows) over multiple robot locations can be leveraged to triangulate the IoT devices. As a demonstration, we deploy WiROS’s receiver on the robot and collect CSI from 3 WiFi transmitters deployed in a 10×5 m environment. Next, by utilizing the bearing estimates provided by WiROS, we can localize these devices with a location accuracy within 2 m. The real-time operation of this case study is provided as a supplementary video³.

V. DISCUSSION AND FUTURE WORK

WiROS’s primary deliverable is to make accurate WiFi-based sensing widely accessible for robotics use cases. Consequently, this work largely focuses on furnishing the calibrated physical WiFi channel measurements and bearings as a key feature to extract from these measurements. These WiFi-based bearings can be applicable for correcting errors accrued in online SLAM, re-localizing robots after sudden changes to their locations, or localizing WiFi transceivers in the environment for IoT management. However, the larger motivation of this work is to facilitate exploring the advantages of RF-based sensing within robot systems. This could be for robot exploration, collision avoidance, or motion planning. Additionally, we would like WiROS to serve as a template for future networking researchers building CSI-extraction tools

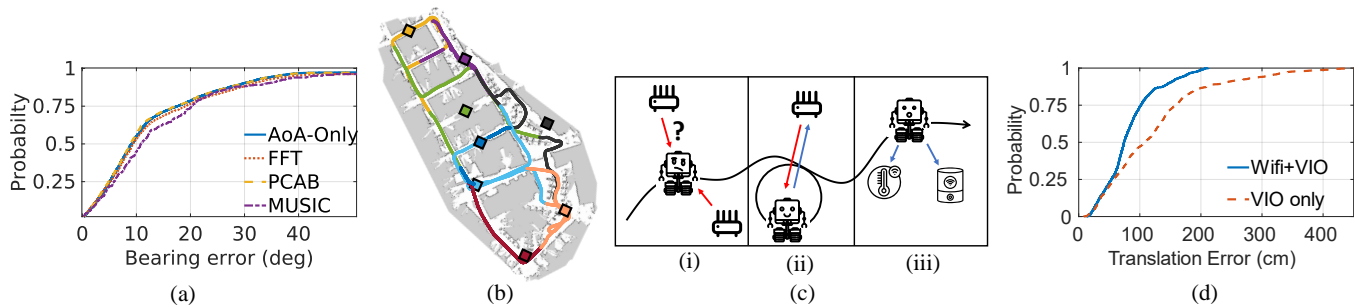


Fig. 8: (a) Bearing estimation accuracy across algorithms, (b) WiROS can automatically detect and lock on to the nearest AP (labeled by color) while exploring the environment. (c) **Case-study overview:** (i) Lost robot can leverage WiROS to localize itself using the AP's in the environment. (ii) Drift during loop closures can be corrected using two-way bearings [1]. (ii) IoT devices can be localized in the environment by leveraging WiROS. (d) Absolute Trajectory error when incorporating WiFi-bearings compared with Kimera [30]

to continue to provide support for newer WiFi protocols, with an immediate future step of extending support to upcoming 802.11ax WiFi protocols [12], [13].

REFERENCES

- [1] A. Arun, R. Ayyalasomayajula, W. Hunter, and D. Bharadia, "P2slam: Bearing based wifi slam for indoor robots," *IEEE Robotics and Automation Letters*, 2022.
- [2] T. Boroushaki, L. Dodds, N. Naeem, and F. Adib, "Fusebot: Rf-visual mechanical search," *Robotics: Science and Systems 2022*, 2022.
- [3] L. Clark, J. A. Edlund, M. S. Net, T. S. Vaquero, and A.-a. Agha-Mohammadi, "Propem-I: Radio propagation environment modeling and learning for communication-aware multi-robot exploration," *arXiv preprint arXiv:2205.01267*, 2022.
- [4] H. Zou, C.-L. Chen, M. Li, J. Yang, Y. Zhou, L. Xie, and C. J. Spanos, "Adversarial learning-enabled automatic wifi indoor radio map construction and adaptation with mobile robot," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6946–6954, 2020.
- [5] P.-Y. Lajoie, S. Hu, G. Beltrame, and L. Carlone, "Modeling perceptual aliasing in slam via discrete-continuous graphical models," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1232–1239, 2019.
- [6] Z. S. Hashemifar, C. Adhivarahan, A. Balakrishnan, and K. Dantu, "Augmenting visual slam with wi-fi sensing for indoor applications," *Autonomous Robots*, vol. 43, no. 8, pp. 2245–2260, 2019.
- [7] Intel, "RealSense D415," <https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/Intel-RealSense-D400-Series-Datasheet.pdf>, 2019, accessed: 2022-03-23.
- [8] "Hokuyo lidar," <https://autonomoustuff.com/product-category/lidar/hokuyo-laser-scanners/>, accessed: 2020-10-31.
- [9] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Tool release: Gathering 802.11 n traces with channel state information," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 1, pp. 53–53, 2011.
- [10] Y. Xie, Z. Li, and M. Li, "Precise power delay profiling with commodity wifi," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '15. New York, NY, USA: ACM, 2015, p. 53–64. [Online]. Available: <http://doi.acm.org/10.1145/2789168.2790124>
- [11] F. Gringoli, M. Schulz, J. Link, and M. Hollick, "Free your csi: A channel state information extraction platform for modern wi-fi chipsets," in *Proceedings of the 13th International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, 2019, pp. 21–28.
- [12] F. Gringoli, M. Cominelli, A. Blanco, and J. Widmer, "Ax-csi: Enabling csi extraction on commercial 802.11 ax wi-fi platforms," in *Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & Characterization*, 2022, pp. 46–53.
- [13] Z. Jiang, T. H. Luan, X. Ren, D. Lv, H. Hao, J. Wang, K. Zhao, W. Xi, Y. Xu, and R. Li, "Eliminating the barriers: demystifying wi-fi baseband design and introducing the picoscenes wi-fi sensing platform," *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4476–4496, 2021.
- [14] E. M.-E. Scott Hassan, "Wifi ddrwt tool," 2010. [Online]. Available: https://github.com/ros-drivers/wifi_ddrwt
- [15] Y. Ma, G. Zhou, and S. Wang, "Wifi sensing with channel state information: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 3, pp. 1–36, 2019.
- [16] N. Jadhav, W. Wang, D. Zhang, S. Kumar, and S. Gil, "Toolbox release: A wifi-based relative bearing framework for robotics," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 13 714–13 721.
- [17] W. Gong and J. Liu, "Roarray: Towards more robust indoor localization using sparse recovery with commodity wifi," *IEEE Transactions on Mobile Computing*, vol. 18, no. 6, p. 1380–1392, jun 2019. [Online]. Available: <https://doi.org/10.1109/TMC.2018.2860018>
- [18] A. B. Pizarro, J. P. Beltrán, M. Cominelli, F. Gringoli, and J. Widmer, "Accurate ubiquitous localization with off-the-shelf iee 802.11ac devices," in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 241–254. [Online]. Available: <https://doi.org/10.1145/3458864.3468850>
- [19] J. Xiong and K. Jamieson, "Secureangle: improving wireless security using angle-of-arrival information," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, pp. 1–6.
- [20] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti, "SpotFi: Decimeter Level Localization Using Wi-Fi," ser. SIGCOMM, 2015.
- [21] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Transactions on Antennas and Propagation*, vol. 34, no. 3, pp. 276–280, 1986.
- [22] Y. Xie, J. Xiong, M. Li, and K. Jamieson, "md-track: Leveraging multi-dimensionality for passive indoor wi-fi tracking," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–16.
- [23] "Asus rt-ac86u router." [Online]. Available: <https://www.asus.com/us/networking-iot-servers/wifi-routers/asus-wifi-routers/rt-ac86u/>
- [24] A. S. Azini, M. R. Kamarudin, and M. Jusoh, "Transparent antenna for wifi application: Rssi and throughput performances at ism 2.4 ghz," *Telecommunication Systems*, vol. 61, pp. 569–577, 2016.
- [25] R. A. Sharma, E. Soltanaghaei, A. Rowe, and V. Sekar, "Lumos: Identifying and localizing diverse hidden {IoT} devices in an unfamiliar environment," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1095–1112.
- [26] C. T. Kelley, *Iterative methods for optimization*. SIAM, 1999.
- [27] O. S. R. F. Inc, "Turtlebot 2," <https://www.turtlebot.com/turtlebot2/>, Turtlebot Robotics, accessed: 2022-1-31.
- [28] "Thinkpad i13 gen 3 intel (13")." [Online]. Available: [https://www.lenovo.com/us/en/p/laptops/thinkpad/thinkpad/i/thinkpad-i13-gen-3-\(13-inch-intel\)/21b3cto1wwus1?orgRef=https%253A%252F%252Fwww.google.com%252F](https://www.lenovo.com/us/en/p/laptops/thinkpad/thinkpad/i/thinkpad-i13-gen-3-(13-inch-intel)/21b3cto1wwus1?orgRef=https%253A%252F%252Fwww.google.com%252F)
- [29] I. C. S. L. S. Committee *et al.*, "Ieee standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11*, 2007.
- [30] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone, "Kimera: From slam to spatial perception with 3d dynamic scene graphs," *The International Journal of Robotics Research*, vol. 40, no. 12-14, pp. 1510–1546, 2021.
- [31] A. P. Neto and F. Tonidandel, "Analysis of wifi localization techniques for kidnapped robot problem," in *2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2022, pp. 53–58.
- [32] Z. Chen, F. Xia, T. Huang, F. Bu, and H. Wang, "A localization method for the internet of things," *The Journal of Supercomputing*, vol. 63, pp. 657–674, 2013.