Radar Based Vital Sign Monitoring With Automated Beam Steering

SP24 Capstone Design: Final Design Report

Team Number SP24-48

Team members: Daniel Gore, Daniel Petronchak, Felipe Valencia, Nithish Warren, Gavin Young

Adviser: Athina Petropulu

***Abstract*—Hospitals commonly monitor an infant's vital signs after birth to quantify the status of its body during incubation. Conventionally, this is done by adhesively connecting biometric devices to every infant in the hospital nursery. By eliminating the need for physical connections and using a single device to simultaneously monitor every infant within a nursery, infants will experience more comfort during incubation, and the cost of vital sign monitoring largely decreases. To monitor vitals wirelessly, a double phase shifter (DPS) phased array will be used to repeatedly steer a beam between desired targets. A plain phased array could be used in this context; however, its ability to target a beam at a precise location is limited. Equipping the antenna array with DPS produces a highly focused beam that can localize closely spaced targets. After targeting the beam, a receiver will record the infant's vitals (heartbeat and breathing rate) over a finite sampling period. To automate this process, we implemented a programmable controller equipped with voltage amplifiers to computationally adjust the control voltages at each phase shifter. Doing this causes a shift in the beam's direction, which depends on the target's location and any unwanted neighboring targets that need to be suppressed (nulls). Finally, using Fast Fourier Transforms (FFTs) and filtering, the frequency information of the infant's vitals can be recovered. To test the effectiveness of this system, we ran an experiment on two human targets. The device successfully recovered each target's vital sign frequency information by automatically steering a beam between them. Our results show that the automated wireless vital sign monitoring system could imply promising applications in pediatric medicine, especially since the system is fully contactless and inexpensive if scaled down in size.**

***Keywords*—Phased array, vital signs, monitoring, automation, wireless.**

I. Introduction

Hospital nurseries commonly monitor infant vital signs by connecting biometric monitors to newborns through their incubators. While this procedure produces accurate vital sign readings, the hardware setup is both costly and potentially uncomfortable for the newborn.

When summing the prices of all the biometric monitors in an incubator and multiplying the result by the number of incubators in a hospital nursery, the net expense can surpass tens or hundreds of thousands of dollars (depending on the size of the nursery). Not to mention, incubator sensors can break over time, so they often need to be replaced which adds additional maintenance expenses. Furthermore, newborns may experience discomfort from the sensors surrounding their body. Between wires posing a risk of entanglement and monitors poking against a baby's skin, the quality of the infant's incubation period may not be ideal. As a result, the infant may become overly stressed and require a nurse to tend to it. This continuous behavior can become exhausting for pediatric nurses, especially if a nursery holds several hundred infants at once.

Therefore, rather than putting a collection of costly sensors in a large array of incubators, it would be more beneficial to have one wireless device that scans infants individually and provides a health report in real time. Such a device would eliminate the potential issues associated with wired monitors, as well as overwhelming costs due to installation, continuous electronic servicing, and occasional replacement.

To develop such a device with current state of the art technology, our team turned to remote sensing through the use of radar [1, 2]. Phased arrays are multi-element antennas that have the capability of steering beams of RF energy (beamformers) in different directions. Depending on the application, beamformers generated by a phased array can be used as an adjustable channel to transmit information to any target inside of it. For example, if a transmitter wanted to send a message to three receivers, and the locations of those three receivers are known to the transmitter, then the transmitter could sequentially steer the beamformer to the three receivers and individually send them a message. However, this works under the assumption that the receivers are spaced very far apart from each other. In the situation where targets are closely spaced together, the region of the beamformer will likely overlap among multiple targets, making one-on-one communication between the transmitter and receiver difficult to isolate.

For the purposes of a nursery, we can assume that incubators will be spaced closely together due to indoor capacity requirements for hospitals. As a result, a highly focused beam must be generated to ensure stable communication between infants and the transmitter. Most plain phased arrays are unable to accommodate such a requirement, which drives the need for a new technological implementation. To combat this problem, our team turned to double phase shifter (DPS) technology [1].

Typically, phased arrays have one phase shifter per antenna

element to shift the beamformer. The transmitter can preset the phase at each phase shifter to pull the beamformer in a certain direction, while also distributing power within the beam so that the side lobes are suppressed. For targets that are far apart from each other, one phase shifter per antenna is enough to establish isolated communication between one transmitter and one target. However, since multiple targets will be spaced closely together for our applications, we needed a solution that condenses the power of the beam into a narrow region. This is where DPS comes in: by adding two phase shifters at each antenna, the distribution of power within the beam becomes much narrower. Figure 1 exemplifies this process, where one-on-one communication has a higher achievable likelihood among closely spaced targets.

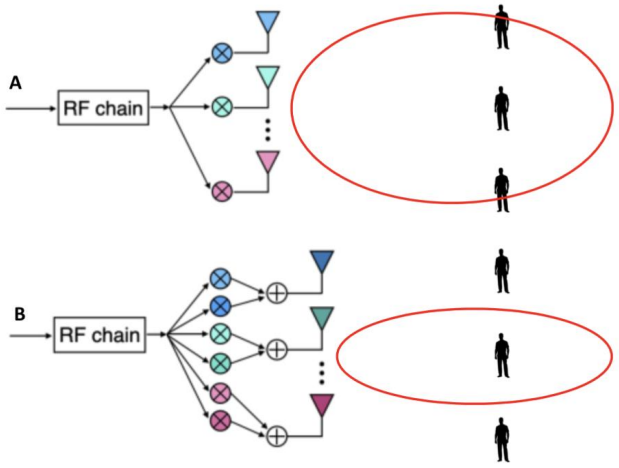

Fig. 1: (a) Example beam of a plain phased array on closely spaced targets. (b) Example beam of a DPS phased array on closely spaced targets.

To adopt this technology into an automatic and wireless infant monitoring system, multiple contributions need to be made. Firstly, with the help of Dr. Michael Wu's group at Rutgers University, we were able to integrate a custom 4-element DPS phased array into our system design. Then, using the mathematical developments in [1], we designed a system to automatically control the two phase shifters at each antenna element of the DPS phased array. The system design is proposed in section 2 of this report, where aspects of hardware and software development, as well as experimental results, are discussed in detail. Section 3 then describes the costs and sustainability of our system, including the financial benefits of downsizing and mass production. Finally, section 4 concludes this report by discussing the overall sensibility of the monitoring system and its future direction.

II. Methods and Results

To monitor an infant's vital signs wirelessly, a variety of hardware and software considerations must be made. In this project, three main questions regarding hardware were scrutinized. Firstly, what kind of controller can we use to automatically steer the beamformer? Since the phased array has 8 total analog phase shifters, our controller must be able to support both analog and digital external subsystems. For this reason, options such as an internal DAC (digital to analog conversion) or PWM (pulse width modulation) must be considered when picking a controller.

Secondly, what kind of driver circuitry is required between the controller and the phase shifters? Most controllers nowadays operate on 5V logic (or less), and they are unable to produce very much power output. This is problematic because the phase shifters being used in the DPS phased array require voltages within 0V to 15V DC. Therefore, a DC voltage amplifier between the controller and phase shifter inputs is necessary if the controller wants to digitally adjust the beamformer.

Finally, how can we recover the vital sign information from the infants? The DPS phased array acts as a transmitter, which will direct a 2.2GHz pulse at each target within the beam. The pulse will then reflect off the target, measuring any vibrations (or motion) occurring within the human subject. Only two vibrations should be detected in this case: the respiration rate and the heart rate. Since the phased array only acts as a transmitter, it cannot detect these vibrations. Therefore, an external receiver board must be configured to recover such information in real time before sending it off to a post-processing backend for vital sign quantification.

At the software level, another two questions are left to be explored. Firstly, how does the phased array know where to steer the beam? To steer the beam itself, the controller will need to change the DC voltage at all 8 phase shifters. However, each voltage must be meticulously calculated via the algorithm in [1], making its software implementation a crucial component of whether or not the system works.

Secondly, how can we process the vital sign information after recovering it? One can imagine that external noise and other factors might cause interruptions during data recovery, so signal processing techniques should be employed to recover each target's heart rate and respiration rate accurately.

With all five of these questions in mind, we developed the following system design in Figure 2. The DPS phased array acts as the transmitter, which localizes a target and transmits a 2.2GHz pulse at them. The receiver recovers the pulse, as well as any vibrations caused by the stationary target. The recovered data then gets sent into a computer, which runs an FFT to determine the frequency components of the receiver data. We expect each FFT to contain two peaks: one corresponding to the target's heart rate and one corresponding to the respiration rate. From there, a filtering stage is added to remove any DC components or other potential targets before displaying the vital sign information of the corresponding target. Finally, once the vital sign information of the first target is recovered, the controller (an Arduino Mega 2560) will determine which target to monitor next and steer the beamformer accordingly. To do this, the controller will send 8 analog voltages into an amplification stage, which will act as drivers for the phase shifters in the DPS phased array.

The following subsections of this report scrutinize the hardware and software designs of the subsystems in Figure 2. Section A describes the design and integration of the voltage

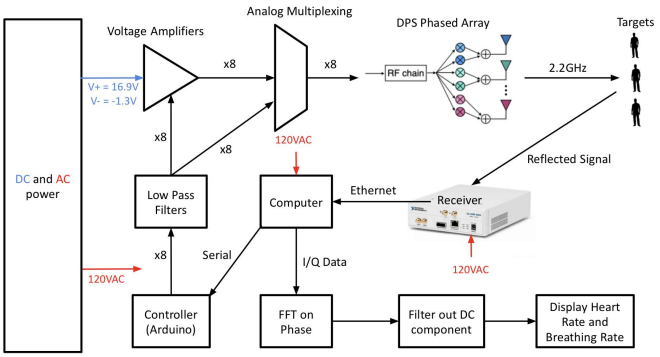

Fig. 2: Proposed solution for wireless vital sign monitoring.

controller and the DC amplifiers for communication between our computer and the DPS phased array. Section B describes the receiver we used for data recovery and the corresponding configuration to set it up. Section C describes the software on the controller, including a voltage configuration algorithm and pin initialization/PWM control code written in Python, and communication code between Python and C++ to allow Python's computations to directly control the pins on the Arduino. Section D will describe the post-processing software used to extract the respiration rate and heart rate from the receiver. Finally, section E will detail the results of tests conducted with the monitoring system using two human targets.

A. Controller and amplifier design

To design a voltage controller that would automatically steer the beamformer by changing the 8 analog phase shifter [3] voltages, we needed to understand the operational requirements of the phase shifters. Fundamentally, each phase shifter contains three ports: "RF IN", "RF OUT", and "Voltage Control". An RF signal $v(t) = Asin(\omega t - \phi)$ is fed into the input port "RF IN", where A is the signal amplitude, $\omega = 2\pi \times 2.2GHz$, t is time, and ϕ is the phase of the input. At the control port, an analog voltage V_{ps} between 0V and 15V can be applied to change the phase of $v(t)$. As a result, $\phi = \phi(V_{ps})$ at the output port "RF OUT", where $v_{out}(t) = Asin(\omega t - \phi(V_{ps}))$.

Since $0V \leq V_{ps} \leq 15V$ while most microcontroller (analog) outputs V_c can only produce $0V \leq V_c \leq 5V$, we needed to design an amplifier stage between the controller and phase shifters to produce a DC voltage gain of at least $\frac{15}{5} = 3$. Furthermore, because each phase shifter has one voltage control port and therefore requires its own amplifier, a total of 8 amplifiers and 8 analog voltage sources must be implemented. For this reason, we decided to use the Arduino Mega 2560. The Mega 2560 is the only Arduino model on the market with at least 8 PWM (analog output) pins, making it the ideal choice for our prototype controller. Of course, instead of using a single Arduino board, we could have used multiple higher-quality controllers (i.e. higher processing speed, less power consumption, etc.) at the same time. However, this would introduce synchronization or clocking issues since multiple boards would be running simultaneously, which we wanted to avoid due to our time constraints on the project.

Given the constraints of a variable 5V input supply and the required 15V (maximum) output, we designed the system in Figure 3. Since the analog output of the Arduino's PWM pins is pulsed with a duty cycle κ, we added a low pass filter to convert the pulsed signal into a DC voltage with a value of 5κ.

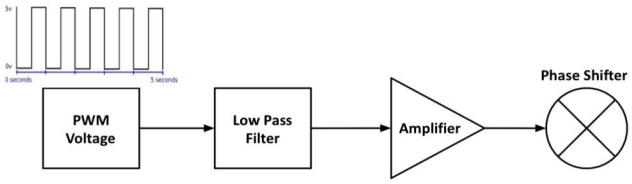

Fig. 3: Voltage conversion from PWM to 0V-15V DC.

After the filter, an amplification stage converts the output range of the PWM pin from 0V-5V to 0V-15V. To do this, we designed an inverting amplifier with a voltage transfer function of:

$$V_{out} = 15 - GV_{in} \tag{1}$$

Where G represents the gain factor of the amplifier. Given an input of 5κ, we get:

$$V_{ps} = 15 - 5G\kappa \tag{2}$$

Furthermore, it is important to note that the Arduino's onboard DAC has an 8-bit resolution. As a result, we are limited to a maximum number of unique voltages that can be produced at the output of the amplifier. To account for this in the above formula, we can define V_{ps} in terms of the DAC's quantization levels a_{in}. For an N bit system, recall $\kappa = \frac{a_{in}}{2^N - 1}$. Then:

$$V_{ps} = 15 - \frac{5Ga_{in}}{2^N - 1} = 15 - \frac{Ga_{in}}{51} \tag{3}$$

Where $N = 8$, $0 \leq a_{in} \leq 255$ and is discretely valued, and the voltage resolution is $\frac{5V}{255} = 20mV$. Since the beam steering algorithm in [1] requires a 200mV resolution at the phase shifter control port, equation 3 is a valid voltage conversion between the Arduino's PWM output and the phase shifter control port. Figures 4 and 5 represent our integration of equation 3 into a hardware setup. The design was implemented onto a breadboard, which was probed to produce the plot in Figure 6.

Our schematic in Figure 4 comes in three stages. The first stage is represented by the 1k resistor in parallel with the 3.3nF capacitor, which is a low-pass filter. The resulting DC output gets passed into the second stage: an inverting amplifier with a transfer function similar to that in equation 1. G is decided by the parallel network of the 1kΩ resistor and the 2.2kΩ feedback resistor, which were hand-tuned to maximize the gain of the amplifier as much as possible. To check the gain during tuning, we probed the red dot in Figure 4 and

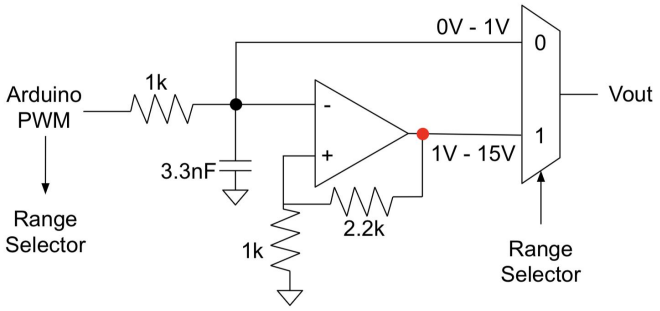

Fig. 4: RC filter and amplifier design with a range extending multiplexer.

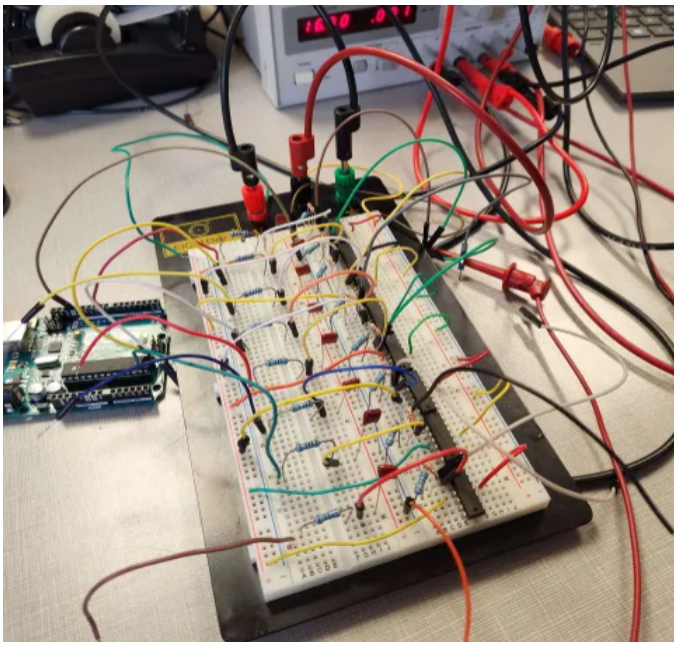

Fig. 5: Implementation of Figure 4 (without multiplexers).

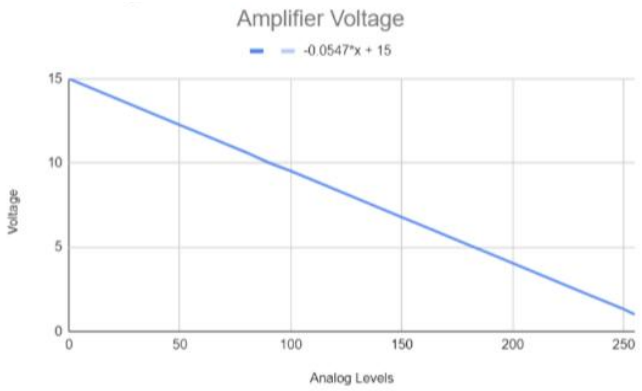

Fig. 6: Plot of phase shifter voltage vs a_{in}.

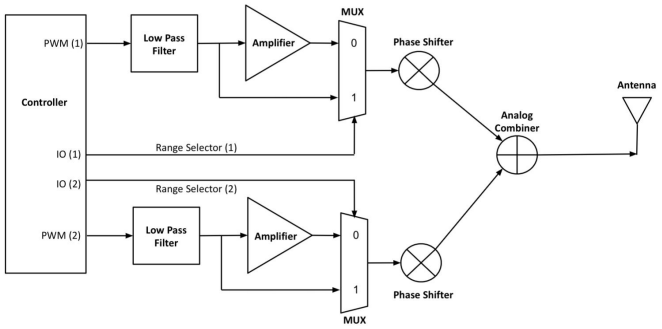

Fig. 7: Voltage controller (for each antenna element).

repeatedly produced V_{ps} vs. a_{in} plots, like the one in Figure 6. In the end, our best outcome was:

$$V_{ps} = 15 - 0.0547a_{in} \tag{4}$$

With a 16.9V positive supply and -1.3V negative supply on the (LF347) operational amplifier [5]. Equation 4 provides the phase shifters with a 1V-15V control range, meaning 0V-1V is not included in the output range. While we tried adjusting the circuit to include the entire range, between adjusting the supply rail voltages and the feedback network, the amplifier would consistently produce output ranges of 1V-15V. These results imply that our chosen chip might be limited in its output range, and therefore requires an external compensation circuit to extend the range.

As a result, a third stage was added to the circuit in Figure 4: a 2-1 multiplexer (MUX) [6]. Since the output range on the RC filter is 0V-5V, we can use digital control to determine which output can be connected to the phase shifter control port. Whenever 1V-15V is required at a certain phase shifter, the MUX will set the amplifier output as the phase shifter control; otherwise, the MUX will bypass the amplifier and form a direct connection between the phase shifter and the filter output. Note that for a majority of beamformers, low control voltages are less common to see. Most of the phase shifters preferably use voltages higher than 1V, so in many cases, the MUX component may be optional. However, for the purposes of this report, we have included every possible outcome of the beamformer and the corresponding hardware.

Now that each phase shifter requires an IO (Input/Output) control pin due to the multiplexer's range selection, on top of a PWM control pin, our controller is required to allocate 17 total pins. 8 pins are used for PWM generation, another 8 pins are used for range selection, and one more pin is required for grounding. Figure 7 describes the overall architecture for the voltage controller hardware, with the assumption that everything in Figures 7 and 8 (specifically the phased array and receiver) is connected to a common ground.

B. Receiver configuration

After our controller tells the phased array to steer a 2.2GHz beam in a unique direction, the signal will reach the stationary target and reflect off of it. As a result, the receiver in figure 8 will be able to pick up the vibrations of any moving object

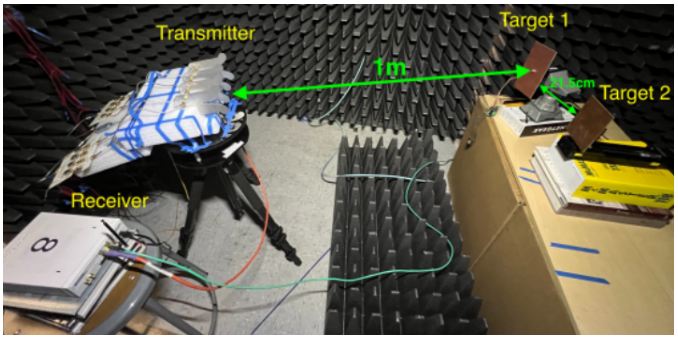

Fig. 8: Example scenario using the same 4-element DPS phased array (acting as the transmitter) in our system and a receiver. For the application of a nursery, the targets would represent stationary infants.

inside of the beamformer, namely a beating heart and lungs. For the purposes of vital sign monitoring, it is imperative for our system to extract these vibrations in the form of frequency information. To do this, we decided to use the USRP 2920 [4], which is a software defined radio (SDR) that can be configured to receive high frequency data and process it at the digital level.

Using LabView, we were able to use the USRP as both a signal generator and a SDR. The USRP has two ports, a TX/RX1 and RX2 port, meaning it has the capability to transmit and receive simultaneously, or receive two different signals simultaneously. For our applications, we wanted to generate a 2.2GHz signal to pass into the phased array, while also reading reflections caused by the beamformer coming into contact with a target. Figure 9 represents the USRP's operation as a subsystem in figure 2, where we use it as a dual signal generator and receiver.

The LabView code we uploaded to the USRP consists of two scripts. The first script is used to generate a 2.2GHz signal through the internal RF front end. The second script implemented a real-time receiver, which could take I/Q (real/imaginary signal) recordings of the surrounding environment and export the data into a csv file. These recordings are taken through the USRP's internal back end, which consists of analog to digital conversion, decimation, downsampling, and numerous filtering stages.

Finally, to configure the USRP's I/Q recording outputs for post processing on our own digital signal processor (a computer), we adjusted several settings in LabView's interface with the device. Our desired I/Q rate was set to 100,000 samples per second. Recordings were allocated to 10 seconds per target, which should be equivalent to the required monitoring time for each target. Therefore, assuming a total of N targets, a recording of one full sweep should be $10N$ seconds and contain a total of $10^6 N$ samples. After every target has been monitored once, the recording data will export into Excel and the USRP will start the next recording. For each exported recording, the post processing steps in subsection D will be taken to extract the breathing rate and heart rate of the desired target.

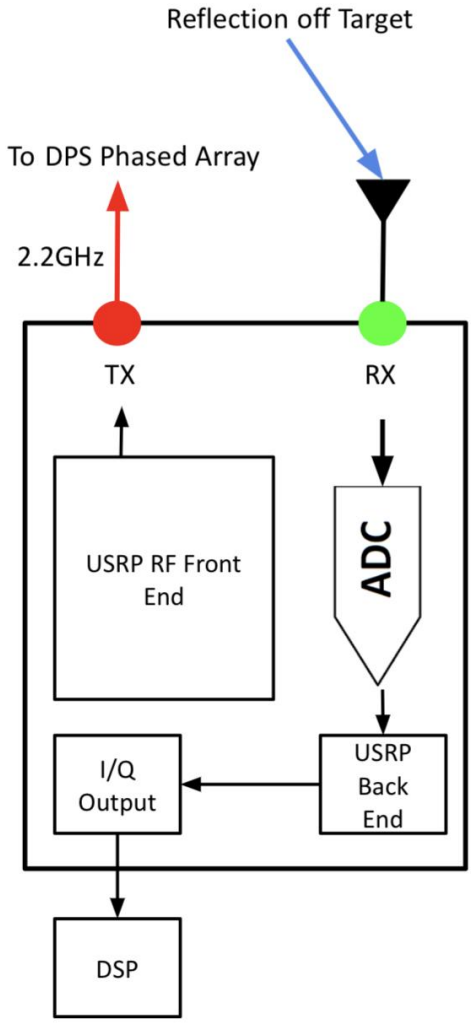

Fig. 9: USRP 2920 acting as a 2.2GHz signal generator and a receiver.

C. Voltage control software

To preset specific analog voltages on the Arduino's PWM pins, we needed to find a way to compute the voltages from a set of input angles. To do this, the Arduino must compile the DPS algorithm presented in [1], which converts a target angle (relative to the center point of the phased array) and a set of nulls into a voltage vector v with 8 entries. If T represents all the closely spaced targets, the algorithm will have to run at least $|T|$ times since each target will have a unique voltage configuration. For the k-th target in T, we can define the target angle as T_k and the nulls as $N_k = \{T_z : T_z \neq T_k\}$ where $z = 1, 2, \ldots, |T|$. Therefore, for every k-th target, the corresponding voltage configuration $v_k = DPS(T_k, N_k)$ where DPS is the algorithm presented in [1].

The main issue with DPS is its computation intensity. Since the algorithm requires a large number of calculations, the Arduino will not have enough memory to support the computation of the voltage configurations. As a result, external software should compute the control voltages, while a different software uses those voltages to set the PWM voltages on the

controller. To accomplish this, we relied on Python's "PySerial" library, which can continuously send data from Python to Arduino via serial connection. A Python implementation of the DPS algorithm would compute 8 voltages in python, before packetizing the results and sending them to the Arduino. The Arduino would then decode the packetized voltages and store them in a vector, which will ultimately be used to control 8 PWM pins and another 8 IO pins in our controller architecture.

Next, we wanted to introduce automated switching functionality to the DPS phased array by making the code multitargeted. To implement this, we can have the user enter multiple target angles in an array T. Since the DPS algorithm in [1] was originally developed to output one set of voltages, we modified it to output a matrix V^* of voltages. For each k-th target, the k-th row of the matrix would equal the corresponding voltage configuration $v_k = DPS(T_k, N_k)$.

$$V^* = [v_1, v_2, \ldots, v_{|T|}]^T \tag{5}$$

The portion of our Python code following V^* involved parsing and processing each v_k to determine the Arduino's PWM outputs. To control the MUX(s) in Figure 4, the Arduino can compute the desired PWM outputs for each voltage in v_k and use IO (Input/Output) pins to drive the digital control pin on the MUXs. To do this, we can use the following conditions:

- If the j-th voltage in v_k is less than 1, the analog level at the corresponding PWM pin is set to: $a_{in} = \lfloor 51 v_{kj} \rfloor$ and the digital pin is set to logic HIGH.
- Otherwise, the PWM pin is set to: $a_{in} = \lfloor \frac{15 - v_{kj}}{0.0547} \rfloor$ and the digital pin goes logic LOW.

This process is conducted for every target before the monitoring period starts. After all the voltage computations are conducted in Python, there is the option to run a beampattern function which allows users to visualize the different beam patterns for each target. This helped us ensure that the optimal voltage configurations were selected during our experiment in subsection E, and the phase shifters were steering the beam in the correct direction.

Once the desired beam patterns are generated, all the pin configurations are set in C++. Then the Arduino steers a beam at the target for a 10 second monitoring period before switching to the next target. This algorithm is fully depicted in figure 10, which visualizes our controller implementation across Python and C++.

The cycle in figure 10 gets repeated infinitely when the device is implemented into a nursery, but during our experiment in subsection E, we limited the number of switching cycles since we just wanted to verify the system's overall operation.

D. Post processing software

After the 2.2GHz signal reflects off the target and passes into the receiver, it is promptly processed through the USRP's digital back end and converted into I/Q data. The I/Q data is then exported into Excel, which is eventually processed (in Matlab) to gather the heart rate and breathing rate of the target.

Recall that I/Q signals represent the real and complex components of a received signal $x[n]$.

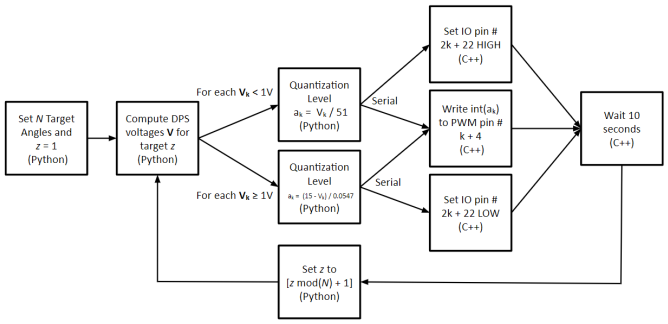

Fig. 10: Voltage control algorithm for the hardware depicted in figure 7.

$$x[n] = I + jQ \tag{6}$$

In order for us to recover frequency information on the I/Q recordings, it is necessary to analyze the phase θ of equation 6.

$$\theta[n] = tan^{-1}(\frac{Q}{I}) \tag{7}$$

Since a Doppler shift is formed by the 2.2GHz signal reflecting off the target, the frequency of vibrations in the target's body will be related to the phase difference of the signals reflecting into the receiver. Therefore, by using the Fast Fourier Transform (FFT) on $\theta[n]$, the resulting $X[\kappa]$ will contain peaks at frequencies corresponding to the target's heart rate and breathing rate.

$$X[\kappa] = \sum_{n=0}^{N-1} \theta[N] e^{\frac{-j2\pi\kappa n}{N}} = \sum_{n=0}^{N-1} tan^{-1}(\frac{Q}{I}) e^{\frac{-j2\pi\kappa n}{N}} \tag{8}$$

Where N is the size of the I/Q recording.

Once the FFT is generated, our Python code will locate the vital sign-related peaks inside the FFT. Since 60 $bpm = 1Hz$ and the high end of heart rate is about 180 bpm, we expect the heart rate peak to be between 1 Hz and 3 Hz. Furthermore, we expect the respiration rate peak to be somewhere between 0.1 Hz and 0.5 Hz. By defining D_H as the domain of heart rates between 1 and 3 Hz, and D_R as the domain of respiration rates between 0.1 and 0.5 Hz, we can determine the vital sign frequencies using:

$$f_H = \text{argmax}_{D_H} \left\{ \sum_{n=0}^{N-1} tan^{-1}(\frac{Q}{I}) e^{\frac{-j2\pi\kappa n}{N}} \right\} \tag{9}$$

$$f_R = \text{argmax}_{D_R} \left\{ \sum_{n=0}^{N-1} tan^{-1}(\frac{Q}{I}) e^{\frac{-j2\pi\kappa n}{N}} \right\} \tag{10}$$

Where f_H and f_R are the heart rate and respiration rate frequencies respectively.

Furthermore, we had to consider the issue of a DC component. Since the USRP's I/Q export considers the entire

frequency spectrum from 0 Hz to >2.4 GHz, we expect to see a large peak at 0 Hz due to the 2.2 GHz signal having a DC offset. When we noticed this in our experiment (in subsection E), an issue immediately arose since we were unable to see the vital sign peaks. Since the DC component is at a much higher power level relative to the surrounding peaks on the FFT spectrum, any surrounding activity gets suppressed, including any peaks within a region of a few Hertz. Therefore, in order to visibly see the vital sign peaks on a spectrum, there is a need to remove the DC component. To do this, we suppressed the magnitude of the peak at 0 Hz by moving it down to the noise floor X_0, such that:

$$X[k = 0Hz] = X_0 \tag{11}$$

By analyzing the resulting FFT, we can determine the frequencies of the recorded vital sign data. For each monitored target, we expect to see 2 peaks: one peak associated with respiration rate on the domain D_R, and another peak (of smaller magnitude) associated with heart rate on the domain D_H.

E. Experimental results

After developing and fully implementing our system, we conducted a real-time test to assess the viability of the wireless vital sign monitor. Figure 11 depicts all the subsystems in figure 2 being connected together, which includes the DPS phased array, the receiver/signal generator, a custom voltage controller, power, and a computer for post-processing.

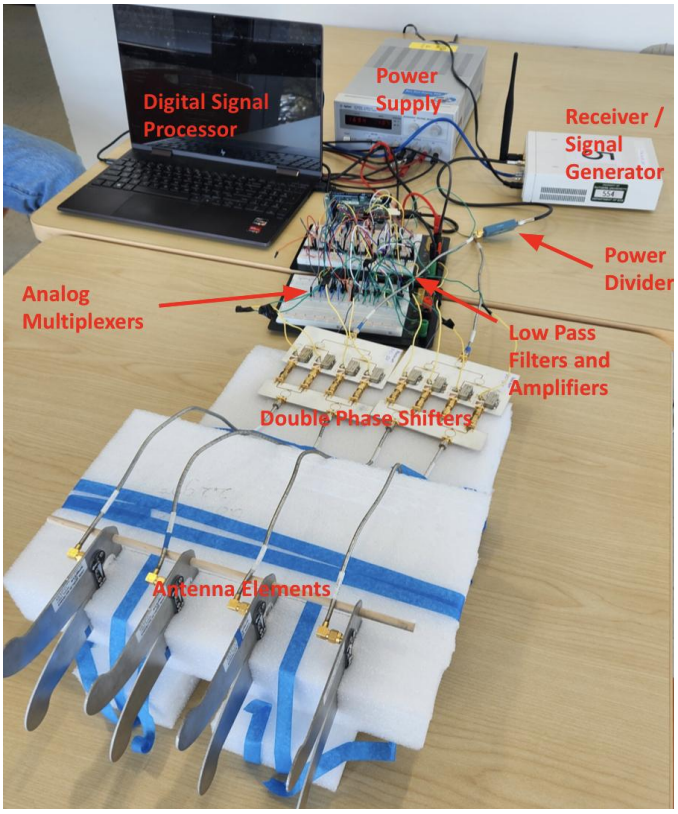

Fig. 11: Full vital sign monitoring system.

To assess the viability of our system, we conducted the test depicted in figure 12 (using IRB protocol HRP-503a). In our experiment, two human targets sat 1.5 meters away from the DPS phased array. One was located $+20°$ off the centerline, while the other target was located at $-20°$. Furthermore, both targets were facing in the direction of the receiving antenna because we wanted to optimize the total number of reflections being directed from the human targets to the receiver. To steer the beam between targets #1 and #2, the control algorithm in figure 10 configured the phase shifter voltages to produce the beam patterns in figure 13.

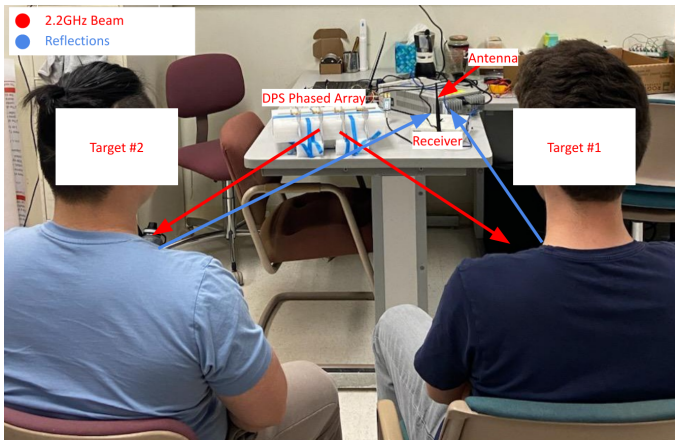

Fig. 12: Test on two human targets.

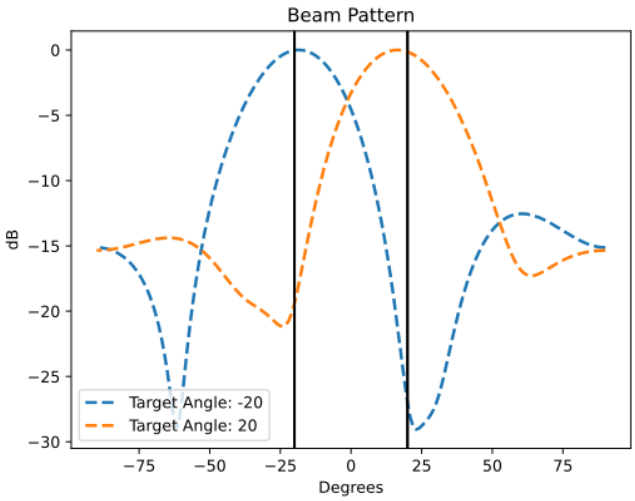

Fig. 13: Beam patterns for targets #1 and #2.

After automatically taking 10 second I/Q recordings of each stationary target and processing the data, the FFT plots in figures 14 and 15 were generated. By analyzing the FFTs, we can see that the breathing rate and heart rate of both targets was successfully recovered. Target #1's estimated breathing rate is 0.24 Hz, while target #2's estimated breathing rate is 0.17 Hz. Furthermore, target #1's estimated heart rate is 1.28 Hz or 77 bpm, while target #2's estimated breathing rate is 1.03 Hz or 62 bpm. Even though clutter is displayed in both plots, the vital signs peaks are noticeably larger, making the vitals

relatively easy to see. These results provided us with insight that the monitoring system can successfully and automatically recover vital sign information of multiple targets.

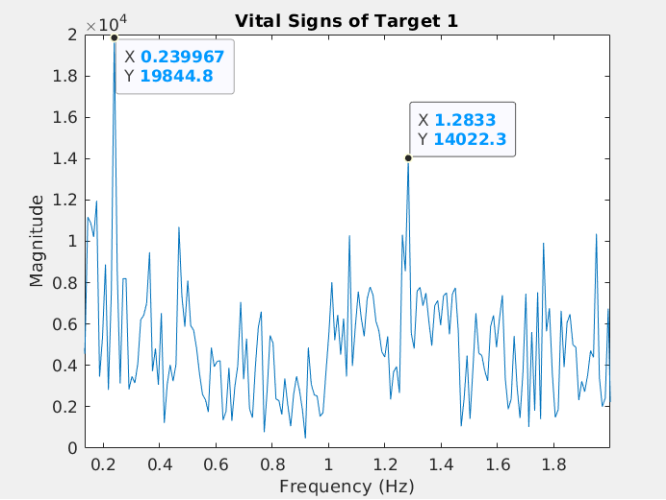

Fig. 14: Frequency spectrum of target #1.

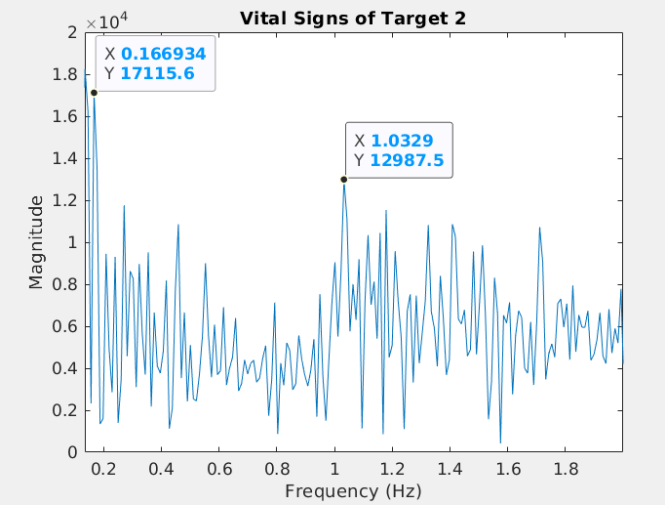

Fig. 15: Frequency spectrum of target #2.

III. Cost and Sustainability

To understand the costs of our prototype, we must first look at each subsystem in Figure 2. The DPS phased array consists of 8 phase shifters, each of which costs around $33, as well as 4 antenna elements priced at $30 each. Furthermore, the prototype consists of circuitry that uses 8 multiplexers (with a price of $5.20 each) and 8 amplifiers (with a price of $0.56 each), all connected to an Arduino Mega 2560 with a price of $42. Our data collection device, the USRP, costs nearly $5500. The hardware also consists of small passive components such as resistors and capacitors, as well as wires, however, the cost of these elements is negligible (under $0.05). Finally, when considering the free components of our project such as a computer, power supply, and ethernet adapter (which we already had available to us), the total estimated cost of the system comes out to just over $6000. A large majority of this total comes from the $5500 receiver (92% of the total), while roughly $500 stems from controller and amplifier circuitry.

Given our prototype costs, it is possible to reduce the overall cost of our system. The clearest method of cost reduction would be to choose a cheaper receiver, preferably one that is small enough to fit onto a printed circuit board (PCB). Changing the controller into an FPGA (Field Programmable Gate Array) would also be beneficial, both in terms of cost (about $10) and processing power (2x or more than the Arduino, depending on the chosen FPGA). Furthermore, in the context of mass production, cheaper amplifier and multiplexer ICs can be used in the design of a single PCB. As shown in Table 1, the overall system expenses (including an external transceiver to replace the USRP) would be around $383, which is a significant price decrease from our current prototype. Even after factoring in the cost of developing a PCB, which typically ranges between $10 to $100 depending on board size, complexity, and other factors, the price differential is defined by a full order of magnitude.

Prototype	PCB Implementation
Phase Shifters: $32 per (x8)	Phase Shifters: $19 per (x8)
Antennas: $30 per (x8)	Antennas: $15 per (x8)
Multiplexers: $5 per (x8)	Multiplexers: $0.38 per (x8)
Amplifiers: $0.56 per (x8)	Amplifiers: $1 per (x8)
RF Transceiver: $5,500	RF Transceiver: $150
Arduino Mega 2560: $42	FPGA: $10 per
Breadboarding costs: $5	PCB fabrication costs: $10 to $100
Rough Total: >$6000	Rough Total: >$400

TABLE I: Differences in cost between the prototype and a PCB implementation.

By compressing our design onto one PCB, a variety of environmental and social benefits are produced. In terms of environmental benefits, fewer materials are needed in the production of a PCB implementation. The prototype relies on large amounts of electronic hardware that stem from heavy mass production, such as the USRP, Arduino board, a computer, and so on (totaling over 5 separate PCBs per unit). By implementing everything onto a single board, a more eco-friendly means of production could be established if the product goes to market. Furthermore, compressing our design into a PCB could cut the power consumption of the system as a whole. Using low-power hardware including FPGAs and chip transceivers over systems like the USRP and Arduino Mega can ultimately save nurseries money. If the system is mass-produced, this could lead to hospitals cutting down on electricity costs by over 20% between our original prototype and its PCB equivalent.

Finally, the leading social impact of the monitoring system will be to positively impact an infant's experience during incubation within hospital nurseries. Current monitoring techniques are reliant on contact, which, as mentioned before, can lead to a poor response from the infant. Even though contact technology is also automated, there is a broad spectrum of inconveniences in utilizing this process. Newborns are sensitive, so contact-based devices can be uncomfortable. Furthermore, newborns can be erratic and may wrap a wire around vital parts

of their bodies causing poor circulation or strangulation. With our prototype being entirely contactless, there is no concern for these potential dangers. As a result, our system could help put nurses under less stress during their nursery shifts, knowing that the infants will likely experience more comfort in their cribs during incubation.

IV. Conclusion

In this report, we analyzed the subsystems of the DPS phased array developed under the supervision of Dr. Michael Wu and its integration into our own monitoring system under the supervision of Dr. Athina Petropulu. Throughout this semester, we were given the opportunity to further develop the phased array and its possible uses in the context of automation. We developed an automatic beam steering algorithm to move the beamformer produced by the phased array. To initiate the algorithm, an operator needs to input only one quantity: a vector containing the desired target angles of the closely spaced infants. Afterwards, an Arduino Mega and additional analog circuitry will drive the phased array to perform one-on-one transmission with each target. A receiver will then recover information produced from the corresponding target and send it to a DSP backend for post processing.

Our system greatly enhanced the ease of use of the DPS phased array since the operator originally needed to do everything manually. Prior to our contributions, if one wanted to use the technology, they would first need to manually determine the relevant voltages for each desired target. Then they would need to manually adjust eight power supplies to set the phase shifter voltages and properly aim the beamformer. Furthermore, by automating the steering capabilities of the device, we gave it the ability of automatic switching. Now that the phased array can automatically switch between multiple targets without the need for human intervention, the monitoring system can collect the heart and breathing rates of multiple people with ease.

Despite the progress we have made throughout the semester, our vision for the DPS phased array has yet to be entirely fulfilled. Not only are we convinced of the practical potential of DPS in pediatric medicine, but we are also advocating for its marketability to the public across various applications once its total cost is reduced. The addition of highly focused beams in phased array technology can contribute to many applications, including sleep apnea studies and physical therapy. Compressing our design onto a PCB would enable these future directions, alongside many others. As our understanding of DPS technology progresses, we hope to continue leveraging its prevalence in the field of wireless health monitoring.

Acknowledgments

We would like to thank Dr. Athina Petropulu and Zhaoyi Xu for their tremendous support and guidance as we worked to implement a prototype for our project. We would also like to thank Dr. Michael Wu, Donglin Gao, and Shuping Li for lending us the DPS phased array and demonstrating how to use it. Finally, we would like to acknowledge the NSF for funding the development of DPS through EECS-2033433.

References

[1] Z. Xu, D. Gao, S. Li, C.-T. M. Wu, and A. Petropulu, “Flexible beam design for vital sign monitoring using a Phased Array equipped with Double-Phase Shifters,” ICASSP 2023 IEEE International Conference on Acoustics, Speech, and Signal Processing.

[2] C. Li, J. Cummings, J. Lam, E. Graves, and W. Wu, ”Radar remote monitoring of vital signs,” 2009 in IEEE Microwave Magazine, vol. 10, no. 1, pp. 47-56.

[3] Mini-circuits, “Phase shifter JSPHS-2484,” https://www.minicircuits.com/ WebStore/dashboard.html?model=JSPHS-2484%2B

[4] National Instruments, “USRP-2920,” https://www.ni.com/en-us/support/model.usrp-2920.html.

[5] Texas Instruments, “Operational Amplifier LF347” https://www.ti.com/product/LF347.

[6] Analog Devices, “ADG419” https://www.analog.com/en/products/adg419.html