Hardware Implementation of a Wireless Backscatter Communication Protocol for Brain-Controlled Spinal Interfaces

Anand Sekar, Laura Arjona, James Rosenthal, Joshua R. Smith, Chet T. Moritz

Abstract—**Brain-Controlled Spinal Interfaces are innovative medical devices which can aid physical rehabilitation to restore volitional movement in areas paralyzed by cervical Spinal Cord Injury. The device entails reading neural signals in the motor cortex, decoding the intention to perform a specific movement from those signals, and stimulating the spinal cord when movement is intended. Decoding the neural signals is computationally-intensive and therefore power-hungry, making portable devices difficult to implement. A wireless implementation of a Brain-Controlled Spinal Interface consisting of an external computer (reader) which utilizes backscatter communication to collect data from an implanted neural sensor (implant) can perform that intensive computation. This thesis focuses on the hardware, implant-side implementation of the wireless backscatter communication protocol on a low-power FPGA.**
Index Terms—**BCSI, BCI, backscatter communication, FM0, neural implant, Verilog**

Acknowledgements

I would like to express my gratitude to Dr. Joshua Smith for overseeing my research and Dr. Chet Moritz for bringing me onto the Restorative Technologies Lab. My sincere thanks to Nicholas Tolley and Soshi Samejima for showing me the ropes with hands-on neural decoding. I would like to recognize the invaluable guidance my mentors James Rosenthal and Laura Arjona have provided through this project.

This research would not have been possible without the support from the Computational Neuroscience Training Grant (CNTG) and the Center for Neurotechnology. I am grateful to Dr. Eric Shea-Brown who introduced me to the extraordinary field of computational neuroscience and persuaded me to apply to the CNTG. I wish to thank the UW CSE Departmental Honors program for motivating this thesis.

I acknowledge the contribution of all the amazing professors at the University of Washington. Heartfelt thanks to my friends and family for supporting me on this journey.

I. Introduction

The paper is organized as follows. §I introduces the problem of spinal cord injury and the concept of neuroplasticity. §II explains how a brain-controlled spinal interface is an effective solution and how each component works. §III covers the various ways brain-controlled spinal interfaces can be implemented. §IV explains the wireless communication protocol and the reader-side implementation. §V covers the implant-side implementation, which is the focus of this thesis. §VI demonstrates the integration of wireless components. §VII theorizes future work. §VIII concludes.

A. Spinal Cord Injury

There are roughly 17, 500 new cases of Spinal Cord Injury (SCI) in the United States each year, primarily caused by vehicular accidents and falls ("Spinal cord injury facts and figures at a glance," 2020). The cervix is the top portion of the spinal cord (containing vertebrae C1-C7) supporting the neck; injuries in this area result in a host of complications, primarily incomplete/ complete tetraplegia/ quadriplegia, i.e. various levels of limited or absent movement and sensation below the neck; the higher the injury, the more severe the effects ("Acute spinal cord injury," 2015). There is no known cure for SCI, however, there are physical rehabilitation interventions which can help patients to recover partial functions. The restoration of upper limb function is the highest treatment priority for improving quality of life for patients with cervical SCI (Inanici, 2018).

B. Neuroplasticity

Donald Hebb wrote a postulate in his book, The Organization of Behavior, which is popularly summarized as "neurons that fire together wire together" (Hebb, 1949). This wiring, also known as Hebbian plasticity or neuroplasticity, takes place during the natural development of the nervous system and during learning, but new research shows that closed-loop, activity-dependent neural devices can strongly influence this wiring (Moritz, 2018). For instance, an artificial connection can be created between two sites in the motor cortex by recording the neuronal activity in one site and electrically stimulating the other site correspondingly; just a couple days of this activity-dependent stimulation resulted in changes which persisted for more than a week (Jackson, Mavoori, & Fetz, 2006).

II. Brain-Controlled Spinal Interfaces

A. BCI to BCSI

This view of neuroplasticity explains how emerging brain-computer interfaces (BCIs) can be used to restore motor function (Tolley, 2019; Capogrosso et al., 2016; Carlson & Millan, 2013; Hochberg et al., 2012; Moritz, Perlmutter, & Fetz, 2008). BCIs are devices which - unlike a keyboard or mouse - extract a user's intention more directly - such as from neural signals - in order to interact with a computer; such technologies can enable users to, for example, control robotic prosthetic limbs (Rao, 2019). Brain-controlled spinal interfaces (BCSIs) could be considered as a subset of BCIs, except the end-effector isn't a computer's cursor or robotic arm, it's - indirectly - the user's own muscles. This indirection is twofold: first, the BCSI stimulates the area of the spinal cord which contains nerves associated with the targeted muscles, not the muscle itself; second, the BCSI only helps the user regain control over the targeted muscles more effectively, it doesn't simply move the muscles as the user intends. The goal of a BCSI is to promote neuroplasticity to restore severed pathways from SCI.

As shown in Figure 1, BCSIs entail (1) recording neural signals from the motor cortex, (2) decoding these signals to predict intended movement, and (3) stimulating the spinal cord surface at the site of injury.

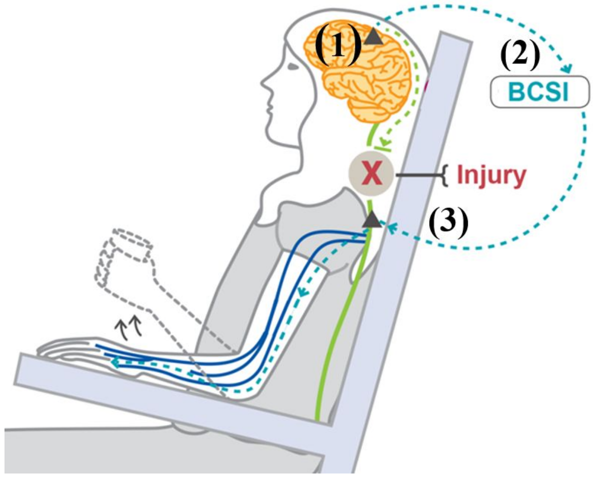
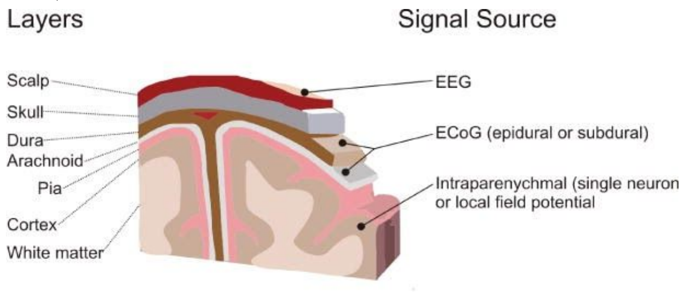


Fig. 1. The three main steps of a BCSI (Moritz, June 7, 2014).

B. Recording Neural Signals

As shown in Figure 2, there are three main methods used to record neural activity: (1) electroencephalography (EEG) with electrodes placed on the scalp, (2) electrocorticography (ECoG) with electrodes implanted over the surface of the cortex, and (3) intracortical electrodes which penetrate the neural tissue (Tolley, 2019). These respectively are more invasive but have a better signal-to-noise ratio. Intracortical electrodes are traditionally used to record the spiking activity of individual neurons, and have worked well for many BCIs; however, local field potentials (LFPs), which represent the sum of synaptic input into neurons, are more stable over time and are also successful at capturing neural signals related to movement intention (Fazli et al., 2009, as cited in Tolley, 2019). The data which our BCSI sensor transmits are LFPs.


Layers
Signal Source
Scalp
Skull
Dura
Arachnoid
Pia
Cortex
White matter
EEG
ECoG (epidural or subdural)
Intraparenchmal (single neuron or local field potential
Fig. 2. Cross-section of the brain representing each of the three common neural recording methods (Leuthardt, Schalk, Roland, Rouse, & Moran, 2009).

C. Spinal Stimulation

Somewhat analogous to recording neural signals, there are three primary methods for stimulating the spinal cord for restoring movement after SCI: (1) transcutaneously with electrodes on the skin above the spinal cord, (2) epidurally with electrodes implanted over the surface of the spinal cord, and (3) intraspinally with electrodes penetrating the spinal cord (Ievins & Moritz, 2017). These are respectively more invasive but have a tighter scope of stimulation.

Regardless of method, there are many aspects of stimulation (electric/ magnetic, frequency, amperage, placement etc…) with some configurations more effective for targeting specific neural pathways than others. Each method of stimulation has been shown to enable control over upper/ lower limbs as well as autonomic (bladder, bowel, sexual etc...) functions (Ievins & Moritz, 2017). Epidural stimulation is effective when applied continuously, whereas intraspinal stimulation requires activity-dependent stimulation to be more effective; these are shown in Figure 3.

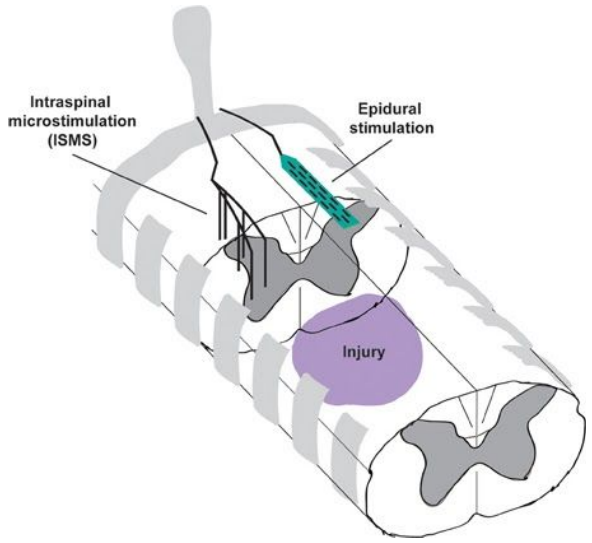


Fig. 3. Intraspinal and epidural stimulation (Mondello, Kasten, Horner, & Moritz, 2014).

D. Neural Decoding

Neurons and neuronal activity encode information: they contain some clues regarding what is happening: sensory input, movement intention, thoughts, etc… Figuring out the relationship between stimuli (events) and neural responses further helps us understand this encoding, which enables us to, for example, decode neural activity to predict when someone wants to move a muscle.

The process of decoding involves identifying which features or aspects of the neural data correspond to an event. In developing a BCSI, we performed experiments with rats. The rats were trained on a lever-pushing task where they touched a nose-poke sensor while pushing a lever. On each successful push, they were rewarded with apple juice. Eventually, they were implanted with a 16-channel intracortical array which recorded LFPs, as seen in Figure 4 (Tolley, 2019).

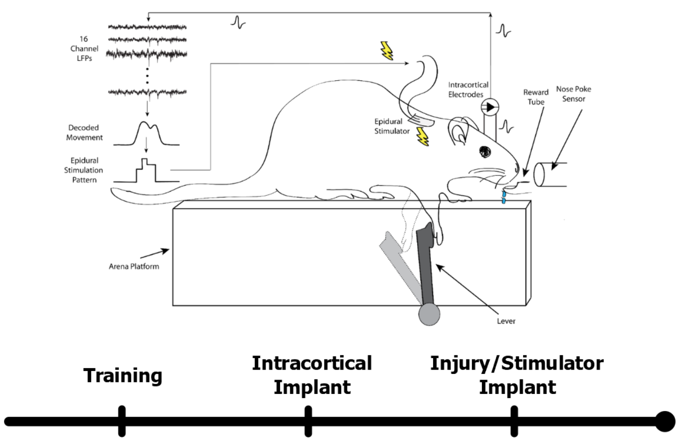


Fig. 4. Rat lever-pushing task (Tolley, 2019).

Thus, we ended up with simultaneous recordings of the neural activity (LFPs) and the event (lever push + nose poke). The data I helped collect from one iteration of the experiment is shown in Figures 5 and 6.

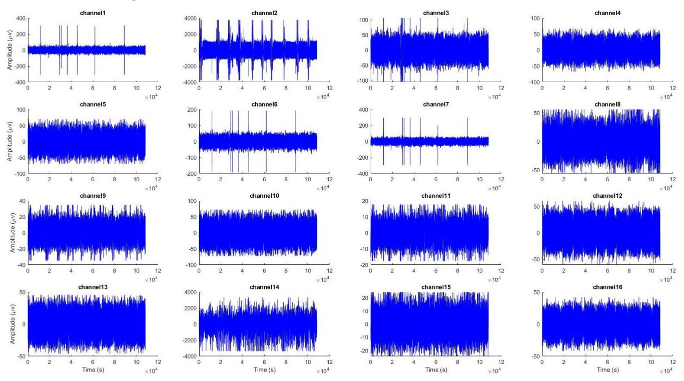


Fig. 5. Amplitude-over-time plots of each of the 16 neural sensor channels. Adapted from code by Nicholas Tolley.

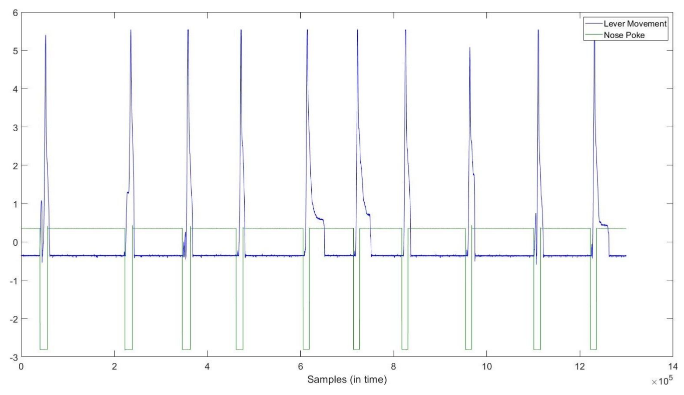


Fig. 6. Nose-poke (green) and lever-push (blue) over time. Adapted from code by Nicholas Tolley.

Khorasani et al. and Tolley et al. found that the feature in LFPs which most clearly correlated with the lever push was the frequency band power in a certain range located between 2 and 900 Hz. To achieve this, we first manually remove any channels which clearly contain too much noise. Then, we pass the raw LFP signals through a common average reference (CAR) filter to get rid of noise common to all channels, followed by a set of band-pass/ low-pass filters and a rectifier, ending up with the power in the desired frequency band over time. The core part of our decoder is a canonical correlation analysis (CCA) filter, which we train with the data. This filter results in a set of coefficients, one for each channel. We can then linearly combine (multiply and add) the weighted band power from each channel to result in a signal which finally predicts lever movement, as seen in Figure 7.

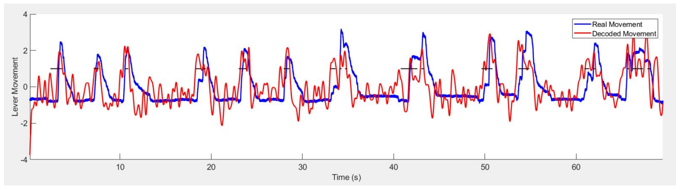


Fig. 7. Real level movement (blue) and roughly decoded lever movement (red). Adapted from code by Nicholas Tolley.

After placing a threshold on predicted movement, we can stimulate the spinal cord whenever that threshold is crossed, i.e. when the prediction is strong enough, resulting in activity-dependent stimulation.

Ultimately, we can consider the BCSI as artificial neurons which promotes a new neural pathway around an injury in the cervical spinal cord which severed a previous neural pathway: decoding one's intention to move, then encoding that intention as stimulation to promote muscle movement.

III. Implementing BCSIs

Through developing the BCSI, the primary way of performing the decoding has been via the TDT (Tucker-Davis Technologies) system - a large, rack-mounted lab computer - as pictured in Figure 8.

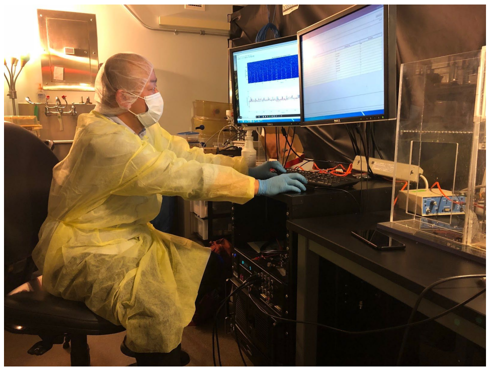


Fig. 8. Soshi Samejima using the TDT (photo taken by Nicholas Tolley).

The development of this device will have to go beyond this medium in order to be mobile. The electrodes used to record the LFPs and the electrodes used to stimulate the spinal cord are already small and relatively power-efficient, i.e. already suitable for mobile use. However, this large computer in-between which takes care of the computationally-intensive (and thus power-hungry) decoding is not suitable for portable use.

One method of implementing the BCSI decoding is via a completely implanted application-specific integrated circuit (ASIC) derived from an FPGA (field-programmable gate array) which can perform the decoding. Such a device has been developed by Ranganathan et al., called the Neural Closed-Loop Implantable Platform (NeuralCLIP) shown in Figure 9. Power supply is already a concern with medically-implantable devices; the main downside of the NeuralCLIP is that it must be carefully and specifically designed to handle the computational load efficiently.

An alternative implementation which circumvents this issue is a wireless one: an external reader utilizing backscatter communication to transfer the data from the implanted electrodes to a computer which can do the intensive decoding. The hardware for such a device has been developed by Rosenthal et al., called NeuroDisc. The tradeoffs between the different implementation approaches are summarized in Table 10.

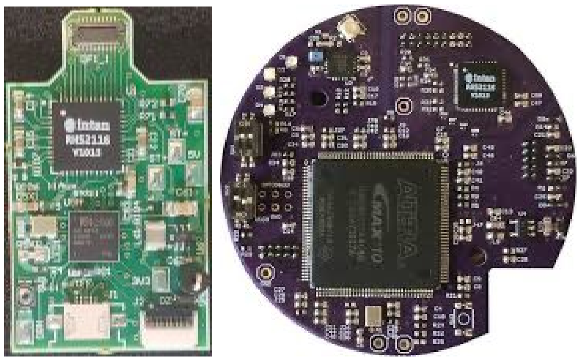


Fig. 9. [Left] NeuralCLIP and [Right] NeuroDisc (Ranganathan et al., 2019; Rosenthal, Kampianakis, Sharma, & Reynolds, 2018).

	TDT	Implanted	Wireless
Computational Resources	High	Low	High
Mobility / Duration	Very Low	High/ Med	High
Battery-dependency	High	High	Low
Communication Latency	Very Low	Low	High

Tab. 10. Comparing different implementations of decoding in BCSI.

IV. The Wireless Protocol

A. Overview

In order to communicate the implanted system with the external reader, a protocol is necessary to specify in what forms information will be exchanged. On a high-level, the protocol involves communication between only two components: the (external) reader, and the (internal) sensor implant. For consistency, we will refer to the following terms from the implant's perspective: "uplink" as information sent from the sensor to the reader, and "downlink" as information sent from the reader to the sensor.

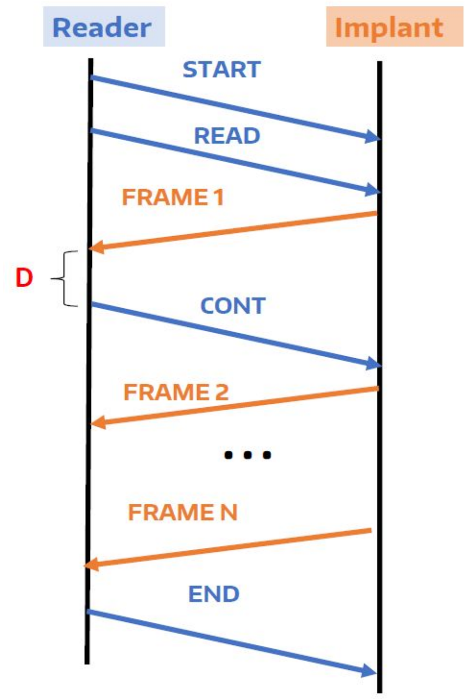


Fig. 11. Sequence diagram of the protocol. START and READ refer to the reader commands, while FRAME contains the transmitted neural data. D refers to the delay from the reader performing feature extraction and responding to the implant.

Figure 11 outlines the protocol. The sensor implant acts like a server and the reader acts like a client: the reader sends "start/read/cont" commands downlink to request information from the implant, and the implant sends back the information uplink in packets (frames). Finally, the reader sends an "end" command to stop the communication.

B. Downlink Communication

The downlink modulation and data encoding works as follows. The reader commands are encoded by Pulse Interval Encoding (PIE), as represented in Figure 12.

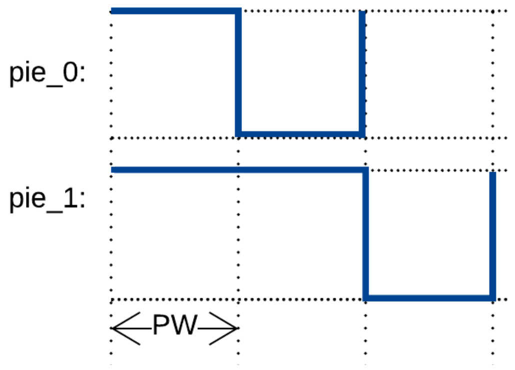


Fig. 12. PIE encoding. PW refers to the duration of the Pulse Width.

The pulse width (PW) corresponds to the duration of half a bit '0'. A bit 0 is represented as high amplitude for one PW followed by low for one PW. It is important to note here that a bit 0 or 1 corresponds to exactly one 0 or 1 PIE symbol. A bit 1 is represented as high amplitude for two PWs followed by low for one PW. Then, the data is modulated by an analog carrier wave via Amplitude Shift Keying (ASK), as represented in Figure 13.

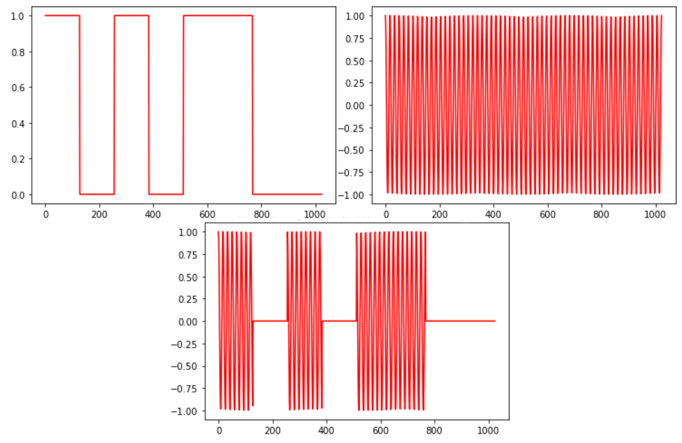


Fig. 13. An example of ASK with on-off keying (OOK). The baseband signal on the left, modulated by the carrier wave on the right, results in the transmitted signal. Adapted from code written by Joshua Smith.

ASK simply oscillates the radio signal at a higher amplitude/power for a bit 1 and at a lower amplitude for a bit 0. PIE keeps the transmitted signal mostly high, with the goal of maximizing the energy flow to power the implant during data transmission. This is the main reason we selected PIE encoding for our reader symbols: since we are using backscatter communication, we want to maximize the energy delivered and available at the implant.

Currently, the PW duration is 6.25µs, making the encoded symbol 0 with duration 2PW = 12.5µs and the encoded symbol 1 with duration 3PW = 18.75µs. Assuming an equal number of 1s and 0s transmitted, this downlink communication achieves a data rate of 64kbps.

There are four commands which are sent: Start, Read, Cont, and End. The Start command consists of the bitstream '00' followed by the 12-bit device ID (to select a particular implant to communicate with). The Read command consists of the bitstream '01' followed by a 16-bit stream indicating which of the 16 channels to transmit and an 8-bit stream indicating the number of consecutive frames to transmit within two reader commands (Frame Count). The Cont command is simply the bitstream '10;' this is sent after every Frame Count number of frames to maintain synchronization. The End command consists of '11' followed by the 12-bit device ID. These are summarized in Table 14.

Command	Structure
Start	[2b: 00 | 12b: device ID]
Read	[2b: 01 | 16b: active channels | 8b: frame count]
Cont	[2b: 10]
End	[2b: 11 | 12b: device ID]

Tab. 14. Reader commands

Each command begins with a frame-synchronization (frame-sync) pulse which is a 12.5μs delimiter followed by a PIE-encoded 0 and calibration pulse, as shown in Figure 15.

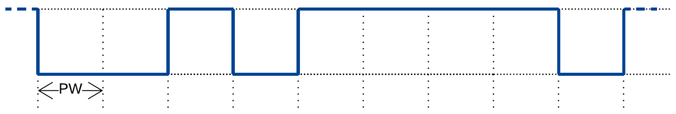


Fig. 15. Frame-sync. PW refers to the duration of the Pulse Width.

C. Uplink Communication

The protocol supports up to 16 neural channels, with 16 bits per channel. After recording the data with the electrodes, the implant encodes the data using Hamming encoding, which is a linear error-correcting code that can detect two-bit errors and correct one-bit errors. The goal of error-correcting codes, like Hamming, is to send redundant information in different ways, such that the data is robust in a noisy environment. Here, we specifically use H(11, 15) which transmits 15 total bits for every 11 data bits.

Next, the reader applies an interleaving algorithm to the frame. The goal of the interleaving process is to increase the likelihood of correction/ detection of burst errors (which become spread). In particular, we use a pattern interleaving algorithm that requires a permutation vector with the same length in bits as the frame. This vector will be predetermined by the reader and implant, so that an adversary would not be able to interpret the intercepted data.

After the hamming and interleaving blocks, the resulting frame bits are encoded via FM0. FM0, also known as differential manchester encoding, is a line encoding scheme which involves a switch on every symbol period. If the bit is 0, it switches again halfway through the period, otherwise if the bit is 1, it stays constant during the period.

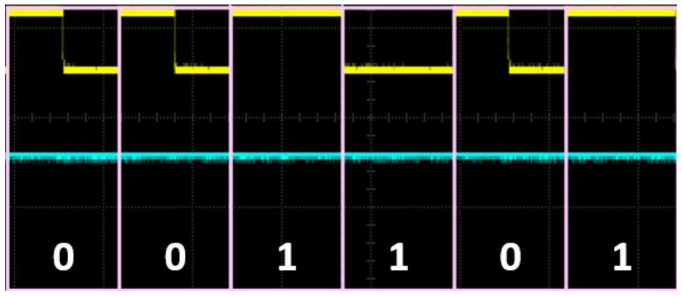


Fig. 16. FM0 encoding (shown by the yellow signal).

As seen in Figure 16, the transitions of the digital signal indicate a logical value (1/ 0), not the value of the digital signal itself (high/ low). One reason to use FM0 modulation to wirelessly transmit the data from the implant is that a transition is guaranteed at every bit boundary, making the synchronization between the implant and the reader easier to achieve. FM0 is also less error-prone in noisy environments than simply comparing the signal levels against a threshold. Moreover, it achieves Zero DC bias, i.e. if the high/ low analog signals are the same magnitude/ opposite polarity, the average voltage is zero, resulting in lower transmitting power necessary and minimal noise (Schouhamer Immink & Pátrovics, 1997).

Finally, the FM0-encoded frame is transmitted using ASK or Phase-Shift Keying (PSK) as Binary-PSK (BPSK) or Differential Quadrature PSK (DQPSK). Quadrature amplitude modulation utilizes complex values to send two orthogonal carrier waves, such that two bits can be sent per symbol, yielding a higher data rate. DQPSK is the method used for backscatter modulation in the NeuroDisc, but our current implementation only uses ASK.

D. Backscatter Communication

One approach for the implant to send data to the reader would be to actively generate and amplify its own carrier frequency. However, this consumes a significant amount of power. Another way for the implant to send data to the reader is for the reader to broadcast a carrier wave, which the implant can passively encode data on by switching the impedance on its antennae, reflecting the carrier wave according to the data. This approach was preferred and used for our system since it consumes significantly less power (Rosenthal, Kampianakis, Sharma, & Reynolds, 2018). The downside is that this results in more path attenuation, i.e. power loss as a wave propagates more distance, which could result in less reliable communication in more noisy environments. To minimize the impact of these possible errors, we used the hamming and interleaving blocks.

E. OSI Model

The Open Systems Interconnection (OSI) model is a hierarchical structure composed of layers which represents how most modern communication infrastructures, such as the Internet, work. This involves several layers of encapsulation - from physical to application - to ensure robust (error-free) communication. The OSI model applied to this protocol is shown in Table 17.

Data Unit	Layer	Description
Data	Application	Implant: Neural signals (LFPs) recording Reader: Neural signals decoding (via GNU Radio)
Data	Presentation	Data representation: Binary 2's complement Data encryption: future work
Data	Session	N/A: The session layer specifies procedures such as restart and termination of operation, hence, there is no need for this layer in our system
Segment	Transport	Controls the reliability of data transfer based on a set of reader commands and implant responses. The reader talks first, and keeps track of the frame counter for frame retransmission
Packet	Network	N/A: The network layer specifies routing with multi-node networks, but our communication is point-to-point
Frame	Data link	Implant: transmits data frames as specified Reader: transmit commands as specified Error correction and detection: Hamming H(11,15) and Interleaving
Symbol	Physical	Wireless (backscatter) communication Implant: 1Mbps, Reader: 64kbps Encoding: FM0 (uplink), PIE (downlink) Modulation: ASK/BPSK

Tab. 17. OSI model applied to the communication protocol.

F. Reader Implementation

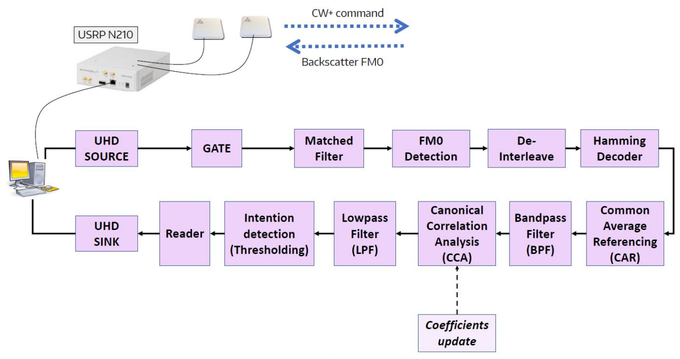


Fig. 18. Communication block diagram for the reader, that can be divided into two parts: received signal decoding and feature extraction.

Figure 18 summarizes the reader-side implementation. The reader uses a Universal Software Radio Peripheral (USRP) software-defined radio to both transmit the Ultra-High Frequency (UHF) waves at 915MHz carrying the reader commands and receive the modulated backscatter signal carrying the implant data. In particular, we use an USRP N210 with an SBX daughterboard. This information is transferred from the USRP to a computer using Gigabit Ethernet which runs GNU Radio. The GNU radio receiver uses a flowgraph approach to process the received data frames and perform feature extraction. The reader can be divided into 4 stages:

1. Signal detection: Gate, Matched Filter
2. Signal decoding: FM0 decoding
3. Error Detection/correction: De-interleaving, Hamming decoding
4. Feature Extraction: CAR, BPF, CCA, LPF, Peak-Finding

Then the reader sends a command back to the implant.

V. Implant Implementation

A. Hardware/ Software Utilized

For prototyping, we use the TinyFPGA BX board, which comprises a 16MHz clock. As mentioned in Section III, we decided to use an FPGA with the ultimate goal of developing a completely implanted ASIC. This FPGA is programmed with Verilog, a hardware description language (HDL). In order to test the Verilog blocks, we used testbenches and the GTKWave software. After simulation, we tested and verified the hardware performance via an oscilloscope.

B. Uplink FM0 Encoding

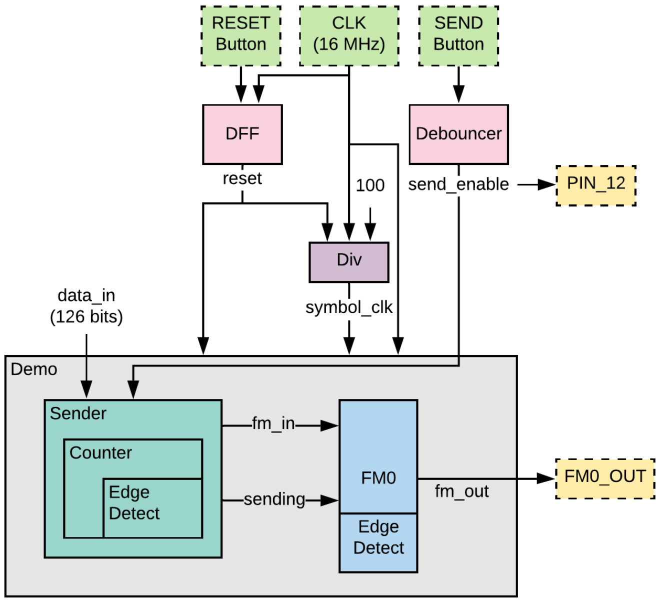


Fig. 19. Block diagram for the FM0 module.

The goal of this module is to take a 126-bit packet (received from the implanted electrodes and neural microchip) and transmit it as FM0 symbols with the proper timing. In the block diagram (Figure 19), each block is a Verilog module and the dashed blocks are hardware-related elements. Every packet corresponds to one 126-bits data frame from the sensor implant.

The communication between the neural microchip and the FPGA is application-dependent. For example, the NeuralClip (Ranganathan et al., 2019) uses Serial Peripheral Interface (SPI) to transfer the data between the implanted neural microchip (Intan Technologies) and the FPGA.

The DFF module is a simple d-flip-flop: it takes in a clock and input signal, and outputs that input signal synchronized to the clock. It ensures that no timing issues could permeate over from hardware. This is used to synchronize the reset signal from the button (Figure 20).

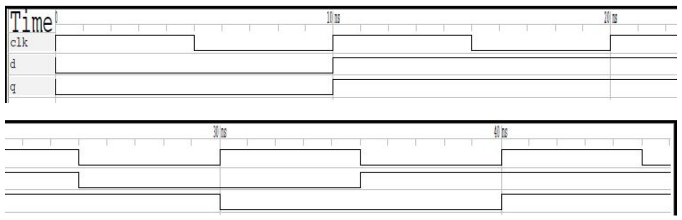


Fig. 20. Simulation of the DFF module. When the input ‘d’ switches on the positive edge of the clock, the output ‘q’ also switches immediately. When the input ‘d’ switches in-between a clock cycle, the output ‘q’ only switches on the next positive edge of the clock.

The Debouncer module takes in a clock and an input signal, and outputs that signal in such a way that removes the bounces caused by flipping a hardware switch. Debouncing ensures that a switch only switches once per flip as intended instead of multiple times in rapid succession. This ensures that when the SEND button is flipped, the ‘send_enable’ signal only has a single positive edge, as opposed to several from bouncing as shown in Figure 21.

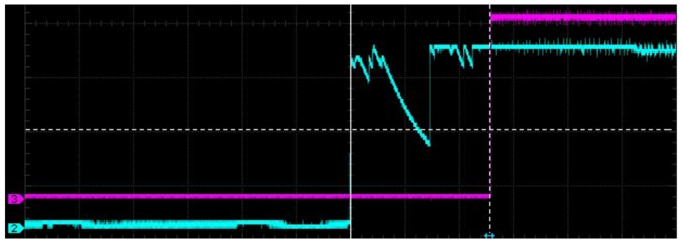


Fig. 21. Oscilloscope screenshot of debouncing signal. The blue (2) signal is the raw signal from the SEND button, which oscillates several times during a single switch flip. The purple (3) signal is the debounced (and delayed) ‘send_enable’ signal which properly flips once after a single switch flip.

The Div module outputs a clock ‘period’ times slower than ‘clk.’ This is used to create the symbol clock, which needs to have a period of 6.25µs, which corresponds to a frequency of 160 KHz. We use this frequency because the implant sensor is required to backscatter the data at 160kbps. The TinyFPGA has a 16 MHz system clock, so we need our symbol clock to be 100x slower than that, i.e. we set ‘period’ to 100. In the simulation (Figure 22), we show how Div can generate a clock that’s 4x slower and 10x slower than the input clock.

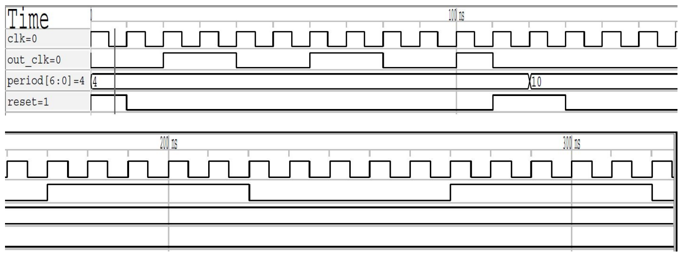


Fig. 22. Simulation of the Div module. When ‘period’ is set to 4, ‘out_clk’ has a period four times as long as ‘clk,’ and when ‘period’ is set to 10, ‘out_clk’ has a period ten times as long as ‘clk.’

The Edge Detect module detects a positive or negative edge from the input. If ‘pos’ is 1, it detects a positive edge, otherwise it detects a negative edge. When an edge is detected, ‘out’ is raised for one clock cycle. This is used in several modules, mainly to detect the positive (or negative) edges of the symbol period. In the simulation (Figure 23), detecting a posedge and negedge occurs respectively.

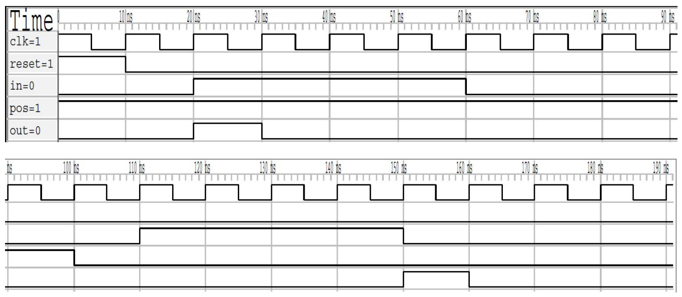


Fig. 23. Simulation of the Edge Detect module. When ‘pos’ is set to 1, ‘out’ becomes high for one clock cycle upon detecting the positive edge from ‘in.’ When ‘pos’ is set to 0, ‘out’ becomes high for one clock cycle upon detecting the negative edge from ‘in.’

The Counter module begins ‘count’ at zero. Once a positive edge is detected in ‘send_enable,’ count begins incrementing every symbol period. Once the count reaches the ‘max’ value, it resets to 0. The ‘sending’ signal is high whenever the module is incrementing and low when it isn’t. This counter is used for indexing into the sensor data, sending it bit by bit. In this case, we’re sending a 126-bit packet, so ‘max’ is set to 125. The ‘sending’ signal is used by the FM0 module to only encode (flip back and forth) if data is being sent, and stay at a flat zero otherwise.

The Sender module sends out each bit of a ‘data_in’ packet, starting when a positive edge is detected in ‘send_enable,’ and sending the next bit every symbol period. It stops sending once the end of the packet is reached. The ‘sending’ signal is high only when data is being sent. The sender simply uses the counter to index through the sensor data. The 126-bit packet is passed as ‘data_in.’ The ‘sending’ signal is passed through from the counter.

Finally, the FM0 module encodes the ‘in’ signal as per FM0, sending according to the symbol period and only sending when ‘sending’ is high. This is shown in Figures 24 and 25 respectively.

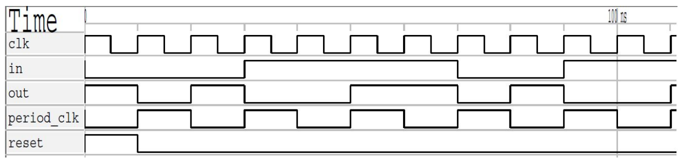


Fig. 24. Simulation of the FM0 module. The ‘period_clk’ is twice the period of the system ‘clk.’ The ‘out’ signal encodes the ‘in’ signal in FM0. The ‘sending’ signal is assumed to be high for this simulation.

The Demo module is a high-level block which simply connects the sender and FM0 modules (Figure 25).

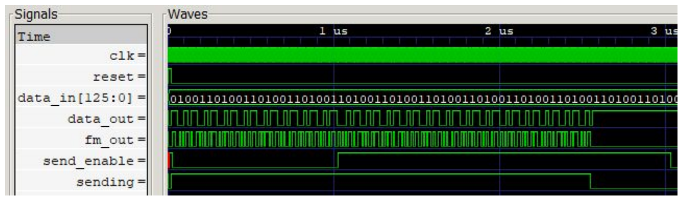


Fig. 25. Simulation of the Demo module. When ‘send_enable’ is first set to high, the packet ‘data_in’ starts being sent bit-by-bit by the sender module through ‘data_out.’ The FM0 module encodes ‘data_out’ through ‘fm_out,’ and only does so when ‘sending’ is high.

The hardware testing setup is shown in Figures 26 and 27. The two switches correspond to the reset and send_enable signals. The oscilloscope’s yellow signal displays the FM0 output.

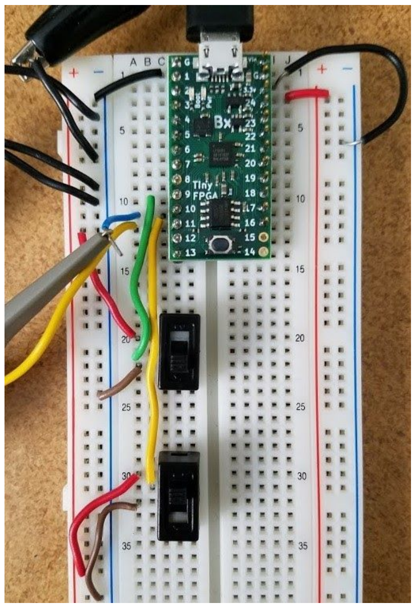


Fig. 26. Hardware testing setup: the circuit. The top button is RESET and the bottom button is SEND. The oscilloscope grounding clip and probe can be seen on the left.

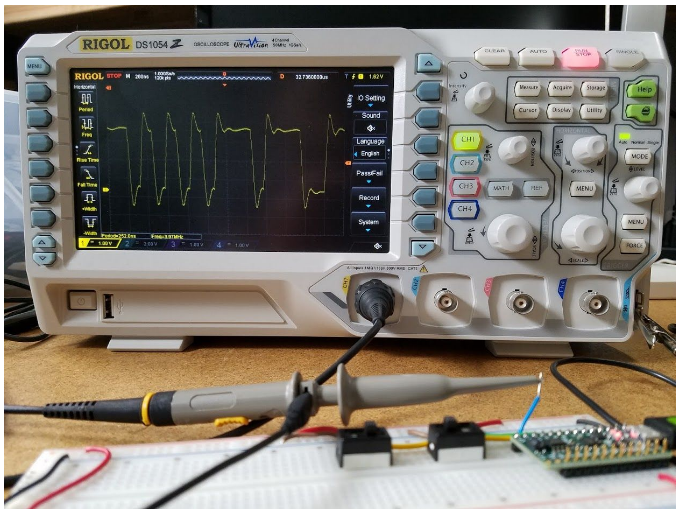
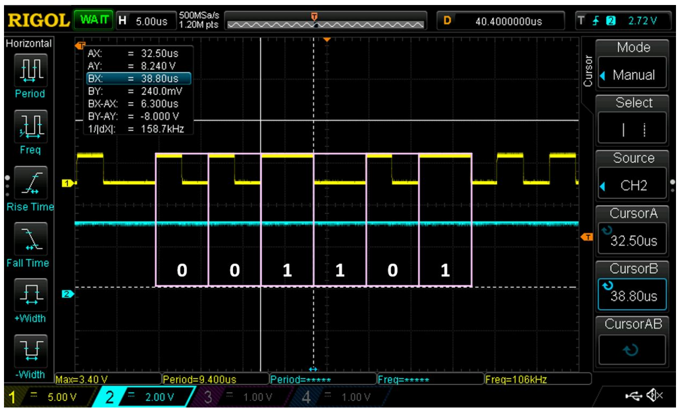


Fig. 27. Hardware testing setup: the oscilloscope. This oscilloscope has multiple channels, enabling us to detect when the ‘send_enable’ signal is triggered and simultaneously see the FM0_OUT signal.

For the sake of testing, the pattern that’s being transmitted is 6 bits repeating: 001101.

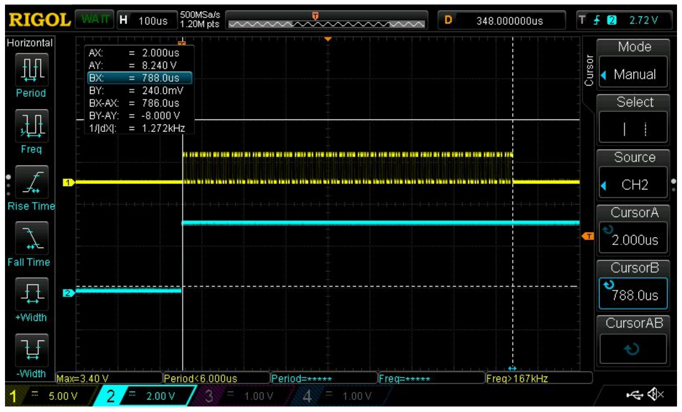


```
AX:     =  32.50us
AY:     =  8.240 V
BX:     =  38.80us
BY:     =  240.0mV
BX-AX:  =  6.300us
BY-AY:  =  -8.000 V
1/|dX|: =  158.7kHz
```

Fig. 28. FM0 as seen on the oscilloscope. The yellow (1) signal is the transmitted signal, and the blue (2) is the ‘send_enable’ signal.

To verify the timing (Figure 28), we used the cursors to see that the BX-AX label equals 6.3000µs and the 1/|dX| label equals 158.7kHz, both of which approximately equal 6.25µs and 160 kHz, our desired symbol period and frequency.

The entire packet is 126 bits, so we should expect the packet length to be $126 \times 6.25\mu s = 787.5\mu s$. We can see in Figure 29 that the BX-AX label properly equals 788.0µs.



```
AX:     = 2.000us
AY:     = 8.240 V
BX:     = 788.0us
BY:     = 240.0mV
BX-AX:  = 786.0us
BY-AY:  = -8.000 V
1/|dX|: = 1.272kHz
```

Fig. 29. A whole FM0-encoded 126-bit packet. The yellow (1) signal is the transmitted signal, and the blue (2) is the 'send_enable' signal.

The reason why the measurements are slightly off is that the granularity at which the cursors move is relatively large: scrolling one unit left or right yields a measurement slightly under or over our expected value.

C. Downlink PIE Decoding

For the sake of efficiency, instead of creating a set of Verilog modules which time and recognize the changes in the input to interpret PIE symbols, we created a correlator. Because the protocol only handles a set of four (sufficiently distinguishable) reader commands, as an initial approach, we can simply correlate the incoming signal with the already known reader commands.

The prototype module takes an input which is the encoded PIE command, and outputs a correlation value with respect to the commands. This module looked specifically for the ReadF command, which consists of frame_sync + 0b'00 0000 0000 0000 0000 0000 1111 (Figure 30).

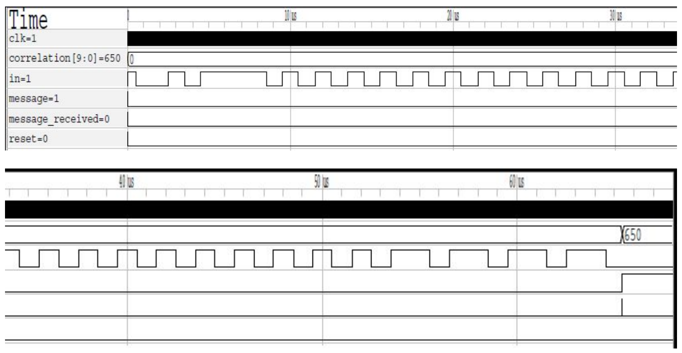


Fig. 30. Decoding the ReadF PIE command.

The module samples every 10 clock cycles for simplicity/efficiency, which makes it such that the number of samples in a ReadF command is 650. The correlation is currently implemented by XOR'ing the sampled input with the command and counting the number of set bits. In the simulation (Figure 30), all the 650 bits match up; hence why the correlation value here is 650.

VI. Integrated Demo

To validate the wireless data transmission, we need to integrate wireless RF connectivity between our FPGA and the reader. To do so, we initially integrated the TinyFPGA with the Wireless Identification Sensing Platform (WISP) (Sample & Smith, 2013). The goal is to control the RF front-end of the WISP with our FPGA. Later on, we want to build an Integrated Circuit (IC) to add the RF front end to our TinyFPGA. James Rosenthal created the following integrated demo, shown in Figure 31, where the WISP - a device which performs the passive, sensor-side backscatter communication - was connected to the TinyFPGA. A level-shifter was required in the WISP Rx path to convert the voltage levels.

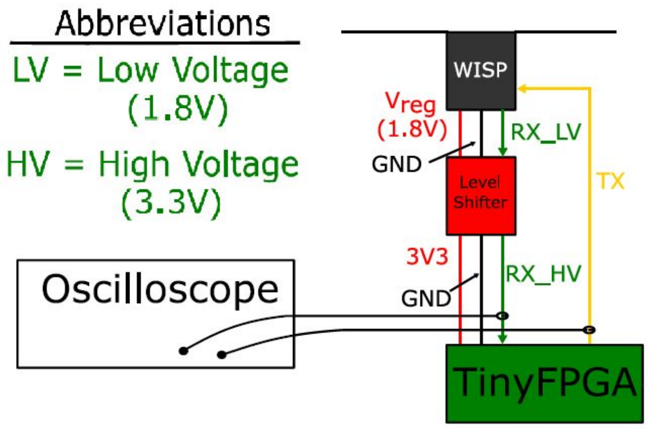
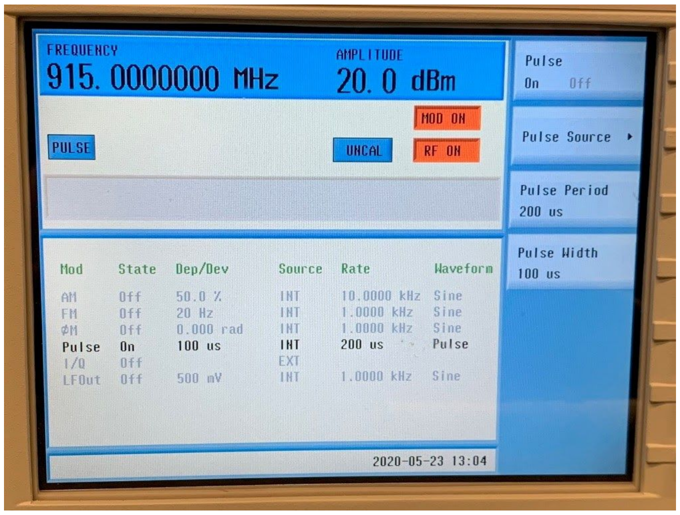


Fig. 31. Diagram of test setup.

To initially validate the RF front-end of our WISP-FPGA setup, James used an RF signal generator that could perform pulse modulation by sending bursts (Figure 32).



FREQUENCY 915.0000000 MHz
AMPLITUDE 20.0 dBm
MOD ON
RF ON
PULSE
UNCAL
Pulse On Off
Pulse Source ▸
Pulse Period 200 us
Pulse Width 100 us

Mod	State	Dep/Dev	Source	Rate	Waveform
AM	Off	50.0 %	INT	10.0000 kHz	Sine
FM	Off	20 Hz	INT	1.0000 kHz	Sine
ΦM	Off	0.000 rad	INT	1.0000 kHz	Sine
Pulse	On	100 us	INT	200 us	Pulse
I/Q	Off		EXT		
LFOut	Off	500 mV	INT	1.0000 kHz	Sine

2020-05-23 13:04

Fig. 32. Signal generator setup. It is set to 915 MHz at 20 dBm, with pulse modulation.

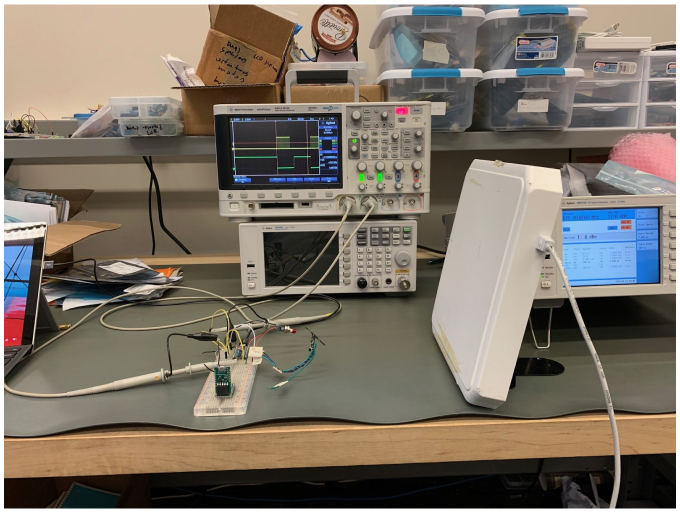


Fig. 33. Overall test setup. The TinyFPGA is connected to a WISP and an oscilloscope. The signal generator is connected to a base station which can emit the backscatter carrier wave.

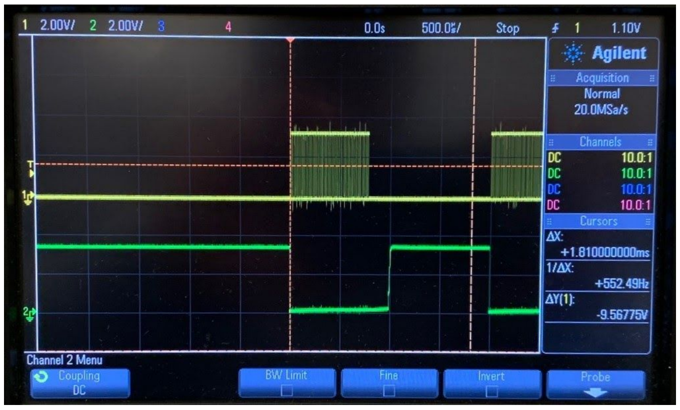


Fig. 34. Oscilloscope showing test results.

The yellow signal in the oscilloscope (Figure 34) is the (FM0-encoded) transmitted packet coming from the FPGA. Green is the mock reader command coming in from the base station (signal generator).

This demo shows two things. Firstly, we successfully used the WISP's RF front-end to receive a digital signal of the correct amplitude in our FPGA. Secondly, the FPGA successfully detected a change in the received signal - emulating a read command - and outputted an FM0-encoded data stream - emulating the implant data frame (Figure 33).

VII. Future Work

Several things still need to be done in order to fully integrate and test all components of this system. After the device has been robustly tested to work with the mock data, we will transmit pre-recorded neural data and test the decoding performance. Once that works thoroughly, we can move forward with live neural decoding.

One important feature which would be necessary for implantation in humans is ensuring the privacy and security of transmitted neural signals. Researchers at the University of Washington have already developed a threat model for BCIs, and have theorized a BCI Anonymzier to prevent malicious attackers from extracting private data and interfering with BCI operation (Bonaci, Calo, & Chizeck, 2015). We could utilize the threat model to develop a similar defense, adding additional encryption and decryption layers into the protocol.

VIII. Conclusion

A BCSI is an effective solution for restoring movement in patients with cervical SCI. We can consider the BCSI as artificial neurons which promote a new neural pathway around an injury in the cervical spinal cord which severed a previous neural pathway: first, decoding one's intention to move, then encoding that intention as stimulation to promote muscle movement. A wireless backscatter medium is an efficient implementation of a BCSI. We've developed and tested components of the hardware implementation of this protocol on the sensor-side: the FPGA can successfully handle both the uplink and downlink communication with the reader. Ultimately, there are several layers of encoding and decoding involved in the communication protocol of the wireless BCSI, much like the lower-level mechanisms of neurons in neural pathways.

Works Cited

Acute spinal cord injury
. (2015). Retrieved from
https://www.hopkinsmedicine.org/health/conditions-and-diseases/acute-spinal-cord-injury

Arjona, L., Rosenthal, J., Smith, J., & Moritz, C. (2019). High performance flexible protocol for backscattered-based neural implants doi:10.1109/APWC.2019.8870386

Bonaci, T., Calo, R., & Chizeck, H. J. (2015). App stores for the brain : Privacy and security in brain-computer interfaces. IEEE Technology and Society Magazine, 34(2), 32-39. doi:10.1109/MTS.2015.2425551

Capogrosso, M., Milekovic, T., Borton, D., Wagner, F., Moraud, E. M., Mignardot, J., . . . Courtine, G. (2016). A brain–spine interface alleviating gait deficits after spinal cord injury in primates. Nature, 539(7628), 284-288. doi:10.1038/nature20118

Carlson, T., & Millan, J. d. R. (2013). Brain-controlled wheelchairs: A robotic architecture. IEEE Robotics & Automation Magazine, 20(1), 65-73. doi:10.1109/MRA.2012.2229936

Fazli, S., Grozea, C., Dan oczy, M. '., Popescu, F., Blankertz, B., & M uller, K. (2009). Subject independent EEG-based BCI decoding. Paper presented at the Proceedings of the 22nd International Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada. 513–521.

Hebb, D. (1949). The organization of behaviour: A neuropsychological theory. New York: Wiley.

Hochberg, L. R., Bacher, D., Jarosiewicz, B., Masse, N. Y., Simeral, J. D., Vogel, J., . . . Donoghue, J. P. (2012). Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. Nature, 485(7398), 372-375. doi:10.1038/nature11076

Ievins, A., & Moritz, C. T. (2017). Therapeutic stimulation for restoration of function after spinal cord injury. Physiology, 32(5), 391-398. doi:10.1152/physiol.00010.2017

Inanici, F., Samejima, S., Gad, P., Edgerton, V. R., Hofstetter, C. P., & Moritz, C. T. (2018). Transcutaneous electrical spinal stimulation promotes long-term recovery of upper extremity function in chronic tetraplegia. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 26(6), 1272-1278. doi:10.1109/TNSRE.2018.2834339

Jackson, A., Mavoori, J., & Fetz, E. E. (2006). Long-term motor cortex plasticity induced by an electronic neural implant. Nature, 444(7115), 56-60. doi:10.1038/nature05226

Leuthardt, E. C., Schalk, G., Roland, J., Rouse, A., & Moran, D. W. (2009). Evolution of brain-computer interfaces: Going beyond classic motor physiology. Neurosurgical Focus FOC, 27(1), E4. doi:10.3171/2009.4.FOCUS0979

Mondello, S., Kasten, M., Horner, P., & Moritz, C. (2014). Therapeutic intraspinal stimulation to generate activity and promote long-term recovery. Frontiers in Neuroscience, 8, 21. doi:10.3389/fnins.2014.00021

Moritz, C. T. (2018). Now is the critical time for engineered neuroplasticity. Neurotherapeutics, 15(3), 628-634. doi:10.1007/s13311-018-0637-0

Moritz, C. T., Perlmutter, S. I., & Fetz, E. E. (2008). Direct control of paralysed muscles by cortical neurons. Nature, 456(7222), 639-642. doi:10.1038/nature07418

Ranganathan, V., Nakahara, J., Samejima, S., Tolley, N., Khorasani, A., Moritz, C. T., & Smith, J. R. (2019). NeuralCLIP: A modular FPGA-based neural interface for closed-loop operation. Paper presented at the 791-794. doi:10.1109/NER.2019.8717135

Rao, R. P. (2019). Towards neural co-processors for the brain: Combining decoding and encoding in brain–computer interfaces doi:https://doi.org/10.1016/j.conb.2019.03.008

Rosenthal, J., Kampianakis, E., Sharma, A., & Reynolds, M. S. (2018). A 6.25 mbps, 12.4 pJ/bit DQPSK backscatter wireless uplink for the NeuroDisc brain-computer interface. Paper presented at the 1-4. doi:10.1109/BIOCAS.2018.8584667

Sample, A., & Smith, J. (2013). The wireless identification and sensing platform. () doi:10.1007/978-1-4419-6166-2_3

Schouhamer Immink, K., & Pátrovics, L. (1997). Performance assessment of DC-free multimode codes. Communications, IEEE Transactions On, 45, 293-299. doi:10.1109/26.558690

Spinal cord injury facts and figures at a glance . (2020). National Spinal Cord Injury Statistical Center, Retrieved from https://www.nscisc.uab.edu/Public/Facts%20and%20Figures%202020.pdf