Wireless Retrieval of a Mouse's Vital Signs via RFID

Daniel Golden
Cornell University

Abstract – Currently, there exists no simple, inexpensive, reliable method of non-invasively obtaining a mouse's vital signs. Most methods involve poking and/or prodding, which tend to excite the mouse, thereby altering pulse and respiration rates from their quiescent values. A novel system is proposed that uses radio frequency identification (RFID) to non-invasively study a mouse's vital signs. The system consists of a base station and a transponder. The transponder is implanted in a mouse, and communicates wirelessly with the base station. The transponder is powered by the low frequency electric field transmitted from the base station, and requires no internal power supply of its own. Code was written in C for the transponder and base station microcontrollers and a Matlab graphical user interface was written for communication from the base station to a PC. Wireless data is transmitted with a Manchester encoding scheme, for which a novel method of software clock recovery is used. The system is not specific to a mouse (though it is designed with one in mind) and can be applied to any small mammal. Currently, the system is not complete, and requires more work to be done before an experimental trial. Electrical, dimensional, and programming design considerations are discussed, with the actual physical effects of the system on the animal left as an exercise for the reader.

I. Introduction

Nobody likes getting shots. The concept of piercing the skin with a long metal tube and injecting fluid directly into the bloodstream tends to give us the willies. There are some who say that they don't mind shots, but they are lying. Everyone minds shots. Even you do.

Often, the mere sight of a needle is enough to make us nervous. Our heart and respiration rates increase, our blood pressure rises, and the hair on the back of our neck stands on end. Ironically, the shot itself ends up being a real let down; it's more the *concept* that we fear.

Imagine that the same procedure, involving an intravenous needle, was necessary to obtain our vital signs: heart rate, respiration rate, blood pressure and body temperature. The stress of the procedure would drive our vital signs up, which would cause them to be always read as unusually high. The results would be meaningless.

This description is likely a good approximation of how a mouse feels when its vitals are being measured. The mouse doesn't know what the experimenter is up to; it could end up being something incredibly painful. As such, mice tend to respond stressfully to any sort of handling that could lead to discomfort. In short, any method of invasively measuring a mouse's vital signs – whether the actual procedure is uncomfortable or not – is, through stressing the subject, prone to distort the very measurements it is intended to obtain.

Clearly, to study mouse behavior, this fundamental problem, hereinafter referred to as the "prod paradox," must be solved. There are two ways to reduce the mouse's stress while measurements are being taken. The first is by sedating the mouse. Though this method is an effective way of mitigating the prod paradox, it still has the undesired effect of altering the mouse's vital signs, this time in the opposite (more sedate) direction. Clearly, as the desired vitals cannot be measured if the mouse is sedated, this method proves rather useless.

The second, more obvious, but more challenging method of circumventing the prod paradox is to measure the mouse's vital signs non-invasively. This involves somehow obtaining the desired quantities without the mouse being aware that measurements are being taken. Traditionally, this task has been unfeasible. However, with the advent of miniaturized electronics and wireless data communication, non-invasive measurements have finally become possible. By implanting a wireless communication-enabled electronic device into the animal, measurements can be taken and sent to a data collection unit, all without the mouse's conscious awareness. This method is usually easier to implement in larger animals, because size constraints are not as severe. However, the very small working space within a mouse necessitates extreme miniaturization. Enter radio frequency identification.

Radio frequency identification (RFID) systems generally consist of two components: a powered base station, and one or more unpowered transponders. By tuning the transponder antennas to the same frequency as the base station antenna, power transmitted by the base station can be captured by the transponder, in essentially the same way that power is transferred in a transformer (Figure 1). This eliminates the need to have a separate power source for the transponder, which in turn allows for miniaturization.

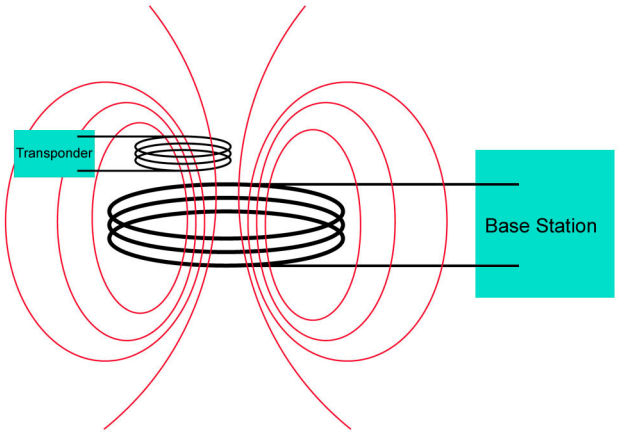

Figure 1. Mutually coupled base station (reader) and transponder (tag) antennas, forming a transformer. Image courtesy [1].

The base station in our RFID system consists of an Atmel U2270B base station integrated circuit (IC), an Atmel Mega32 microcontroller, an inductive coil antenna, and associated circuitry. An RS-232 serial cable is provided to interface the base station with a standard Windows PC running Matlab. Our transponder consists of an Atmel U3280M transponder IC, an Atmel Tiny13 microcontroller, an inductive coil antenna, and associated circuitry. The system operates at a transmission frequency of 125 kHz.

Both the base station and transponder are capable of transmitting and receiving data. The base station transmits data by modulating the magnitude of the electric field sent through its coil. The transponder can detect the modulated field through its own coil to recover the data. In contrast, the transponder does not "transmit" data per se; instead, it systematically damps its own coil's load, which modulates the signal that is reflected back to the base station (Figure 2). This is effectively a passive method of transmitting data. In practice, our system is unidirectional, and only makes use of data transmission from the transponder to the base station.

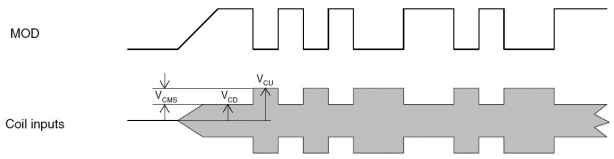

Figure 2. Transponder coil load modulation for data transmission. Image courtesy [3].

This paper describes in detail the complete system, including hardware and software considerations. Sufficient detail is provided so that the user, given commercially available hardware, could reconstruct the entire system for his or herself. As time constraints prevented a final, polished system from being developed, a future work section describes progress that must be made before the system is ready for an actual experimental trial.

II. The Transponder

Our prior experience with Atmel microcontrollers, and the knowledge that Atmel has developed several well-documented RFID ICs, led to the realization of this project. Traditional transponders are made to simply broadcast a serial number, which is useful for identifying a single transponder-tagged object out of many (hence the ID in RFID), but not for much else. As our transponder was required to collect data as well as transmit it – capabilities that are easily realized with a microcontroller – we needed a transponder with a microcontroller interface. Of Atmel's transponder offerings, only the U3280M provided an interface for a microcontroller, making it an obvious choice.

The beauty of RFID is that the transponder needs no power supply of its own; it is powered from the current induced in its coil from the base station's signal. Therefore, the transponder does not need a battery, which is advantageous for two reasons. First, batteries with any reasonable lifespan are often relatively large, and would add significantly to the bulk of our circuit. Second, even a large battery has a limited life span. Replacing a battery after it has been implanted into a living creature is no picnic, neither for the researcher, nor for the animal.

The U3280M is designed such that the power that it collects from its antenna is regulated at approximately 2.7 volts, and can therefore be used to power a low-voltage microcontroller, along with a small amount of associated circuitry. Our low-voltage microcontroller of choice was the Atmel Tiny13V, chosen because of its low voltage requirement (1.8 V minimum) and small form factor (5 x 6 x 1.5 mm). The Tiny13V's internal clock runs at 9.6 MHz. It also has a 10-bit ADC, which is essential for vital-sign acquisition with analog sensors. In our testing, the Tiny13V was operated from an Atmel STK 500 development board, which offered programming capability, convenient access to pins, and a regulated power supply.

The antenna that is used consists of thin insulated wire wound into a coil. The coil is air-cored, and consists of approximately 67 tightly-wound turns in a diameter of about 3.5 inches. For testing purposes, the coil's diameter and number of turns are relatively arbitrary; the quantity of interest is the inductance of the coil, and can be found roughly with the following formula [5]:

$$L(\mu\mathrm{H}) = \frac{d^2 n^2}{18d + 40\ell} \qquad (1)$$

where $L(\mu\mathrm{H})$ is the inductance of the coil in microhenries, d is the diameter of the coil in inches, n is the number of turns in the coil, and ℓ is the coil length (in the direction of the axis around which the coil is wound) in inches. For our coil, if we assume a small (0.25 inch) average length, equation (1) provides an estimate for the coil's inductance of about 750 uH, which is similar to the measured value of 737 uH.

A National Semiconductor LM61 2.7 V temperature sensor is used to simulate the biological information that one would expect to get from a mouse. To reduce noise in the temperature sensor's output, a simple RC lowpass filter with a 0.1 μF capacitor and 1 kΩ resistor is incorporated. This filter can be altered or removed depending on the type of data being acquired. With the filter in place, temperature can be measured with a random error of less than 2°C.

The complete testing circuit is shown in schematically in Figure 3 and pictorially in Figure 4. It is important to note that, in its current form, the transponder is not ready to implantation in a mouse. See the Future Work section for further details.

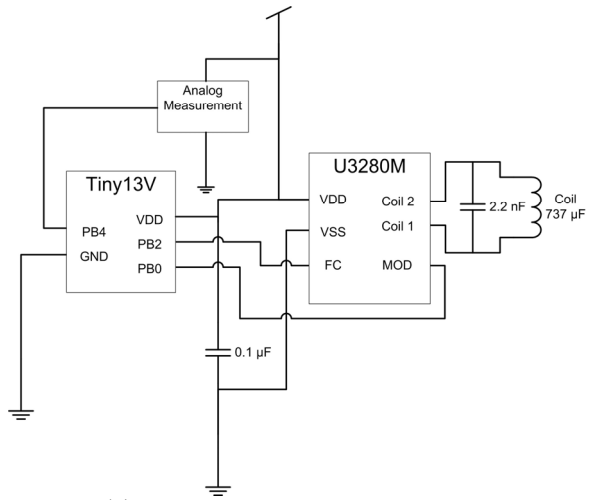

Figure 3. Transponder circuit schematic. Note that there is no separate power source for this circuit; power and ground are provided by the U3280M (though a separate source *was* used during testing).

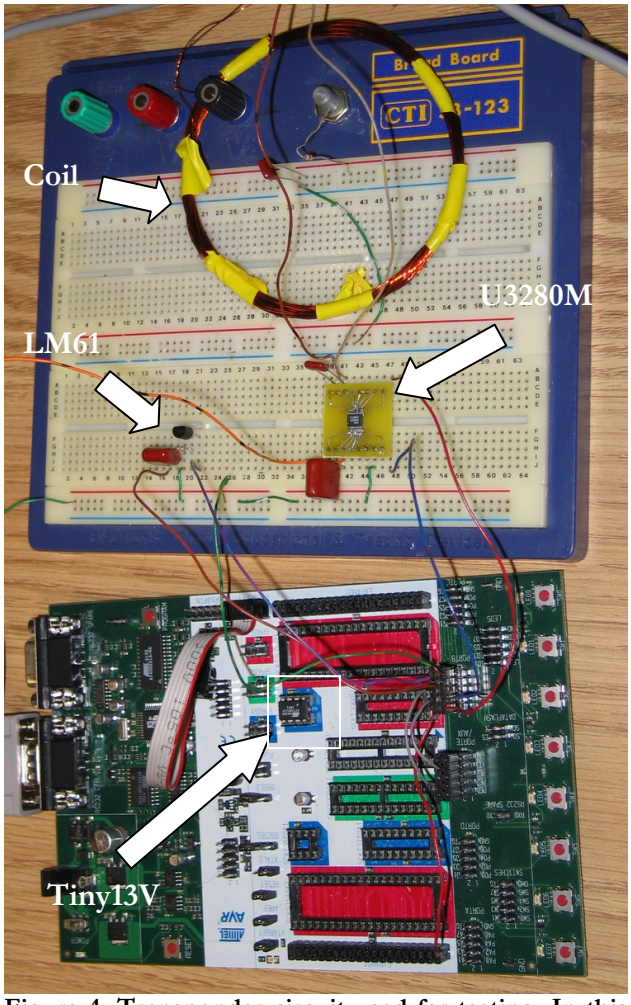

Figure 4. Transponder circuit used for testing. In this incarnation, it clearly would not fit in a mouse.

III. The Base Station

The U2270B seems to be Atmel's only commercially available 125 kHz RFID base station, so that made the choice of base station IC rather simple. The U2270B is paired with an Atmel Mega32 microcontroller, a robust device that we have a lot of experience with. The Mega32 operates at 16 MHz with a 5 V power supply, and has a slew of features. Of particular use is the Mega32's USART, which facilitates communicating with a PC through a standard serial port. As with the Tiny13V, the Mega32 is operated from an STK 500. Here, the STK 500 offers the additional advantage of automatically converting signal voltages from the microcontroller's USART to make them compatible with a PC.

Because the base station is doing most of the complex processing in an RFID system, and because it can be larger, its circuitry ends up being more complicated than that of the transponder. Luckily, Atmel includes several functional circuit designs in the U2270B's data sheet. To save time during the initial phase of the project, we implemented the simplest circuit. Figure 5 shows the circuit schematically, and Figure 6 and Figure 7 show images of the circuit and microcontroller.

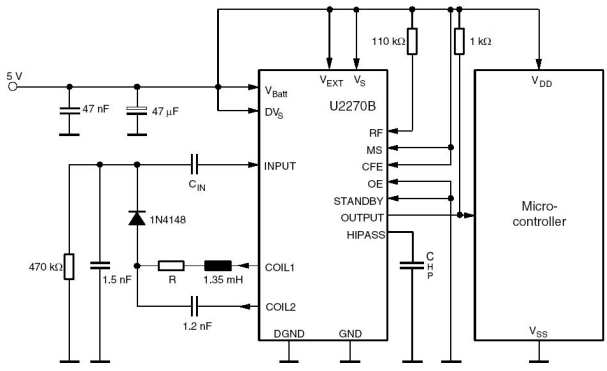

Figure 5. Base station circuit schematic. To achieve a data rate of 1.95 kbps, C_{in} = 1.2 nF, C_{HP} = 220 nF, and R = 100 Ω. Image modified slightly from [2].

The base station coil is wound with the same diameter coil as the transponder, though, to achieve its higher inductance of 1.35 mH, it has

more turns. The trial and error inherent in its design precluded an exact measurement for the number of turns in the coil, but it is estimated to be around 90.

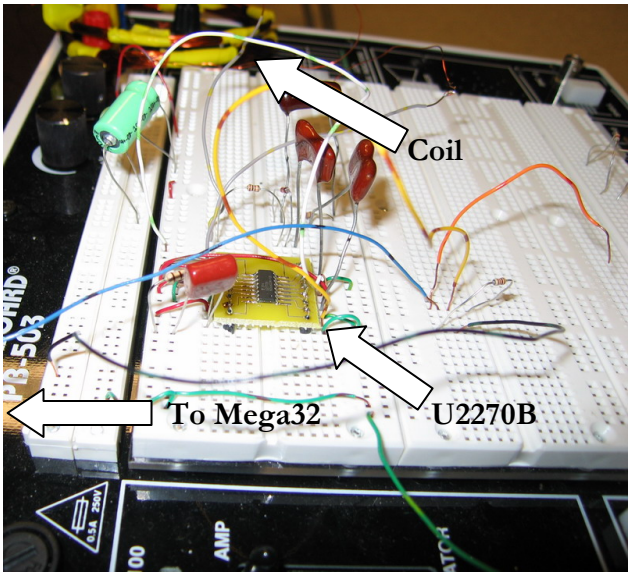

Figure 6. Base station circuit used for testing.

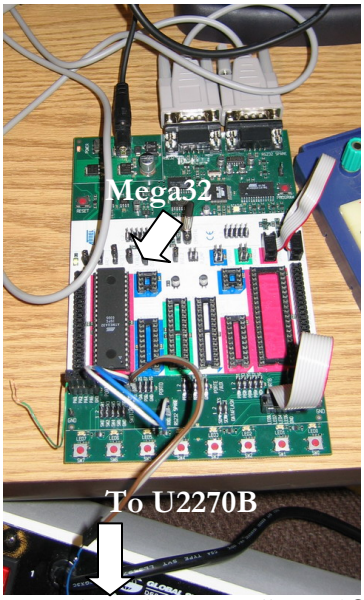

Figure 7. Base station microcontroller and STK 500.

In order to read received data from the base station, data is transmitted serially, via the RS-232 communications standard, to a PC running Matlab on Windows XP. Matlab is used because of our extensive experience with it, in addition to its ease of use and ubiquity in academic institutions. A Matlab GUI was also written to facilitate data reception. Data is transmitted as fast as it is received, at 1.95 kbps, or about 240 bytes per second.

IV. The Matlab GUI

The Matlab GUI (Figure 8) was written in order to provide a user friendly means of acquiring data from the base station microcontroller. The GUI was made using Matlab's GUIDE feature, which offers an intuitive graphical approach to GUI construction.

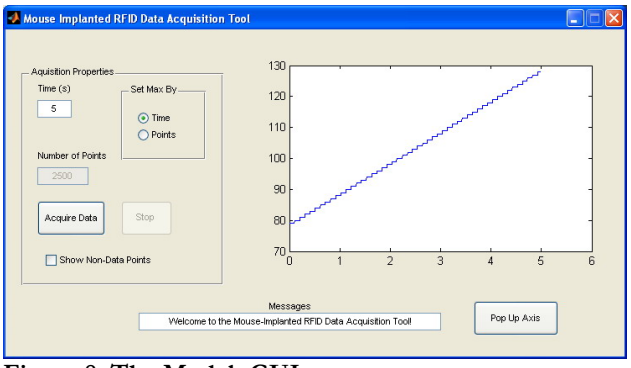

Figure 8. The Matlab GUI.

Via the GUI, the user can choose to acquire data for a certain amount of time, or to acquire a certain number of points. The graph is updated continuously as data is acquired, and data acquisition can be stopped at any time. The program informs the user if there is a problem with serial communication.

If the user is not satisfied with the small graphing window included in the GUI, the Pop Up Axis button plots the data on a traditional Matlab graphing axis, which allows the user to zoom, manipulate the plotting style, etc. The Show Non-Data Points checkbox enables the viewing of acquired data points that do not correspond to valid data; this includes synchronization bytes and invalid data. After the data has been acquired, the data vectors are dumped to the current Matlab workstation, to allow for any additional analyses that the user desires. The Matlab GUI is simple, and it allows for data manipulation in a straightforward and occasionally amusing style.

V. Data Encoding and Decoding

Serial transmission in wired systems generally consists of at least two transmission lines: one carrying the data, and the other the clock to which the data is synchronized. Wireless transmission, however, is an entirely different animal. Wireless data has only one medium to travel through – the air – and as such, cannot support separate transmission of data and clock. As such, traditional non-return-to-zero (NRZ) data, in which a logic one is a high signal for one clock period, and a logic zero is a low signal for one clock period, cannot be used. A data stream, in general, can contain long strings of ones or zeros; these would be represented in an NRZ stream by very long DC values, during which the receiving system may lose synchronization with the transmitter's clock.

To combat this, Manchester code (Figure 9) is used. Manchester code incorporates a transition in the middle of every transmitted bit. A logic one is represented by a transition from low to high, and a logic zero is represented by a transition from high to low. As the transmitted signal must change at least once for every bit transmitted, the problem of transmitting long DC values is eliminated.

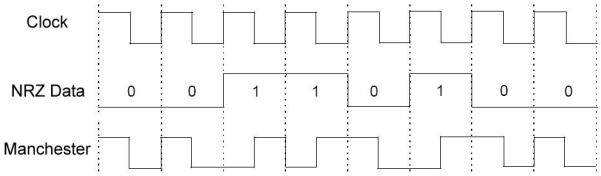

Figure 9. Example of Manchester encoding. Image courtesy [4].

Unlike NRZ data with a separate clock signal, the clock is not provided explicitly to the device receiving Manchester encoded data; instead, it is given, encoded, in the transmitted data. As such, the clock must be recovered from the data. Traditionally, this is accomplished via a phase-locked loop (PLL), a separate hardware component that, via some complicated hardware processes, takes in the received data stream and outputs the transmitter's clock. However, phase-locked loops are expensive, and are unnecessary for this application, especially considering that we have all the power of the Mega32 at our disposal. Therefore, clock recovery is implemented in software on the Mega32.

The software scheme used for clock recovery is relatively simple. At the base station's microcontroller, any data edge triggers a software interrupt, which starts a hardware timer. When the next edge is received, the value of the timer is examined. If the timer's value is equal to one half of a clock period (about 256 μs at a transmission frequency of 1.95 kbps), then the clock remains ambiguous, because an edge may occur either in the middle of a bit, or between two bits. However, if the timer has counted one full clock period (about 513 μs), which can only occur between two bit centers, the clock is recovered, and the current edge is determined to be on the falling edge of the clock (Figure 10). Using this technique, as soon as the first transition between two different-valued bits (which results in a long data edge) is encountered, the clock is recovered.

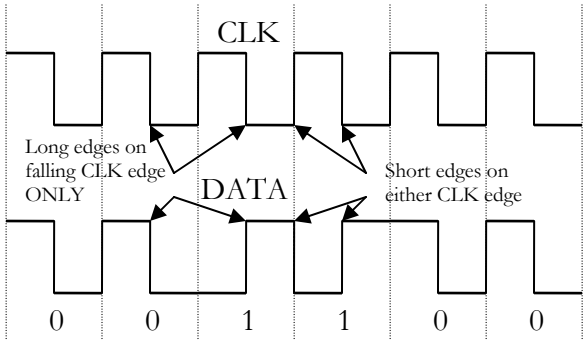

Figure 10. The principle behind clock recovery. A short period between data edges occurs with one edge on a rising clock edge and the other on a falling clock edge – but the determination of which data edge corresponds to which clock edge is ambiguous. A long period between data edges always occurs with both data edges on the falling edge of the clock.

Recovery of the transmitted clock leads directly to the determination of bit boundaries, i.e., determining which edges of the data are between bits and which are in the middle of bits. More complicated is the problem of recovering the boundaries between bytes. Because data is continuously transmitted in this RFID system,

there are no pauses or other signifying boundaries between bytes. For example, the stream 00110001 may be one eight-bit data byte, or it may be the trailing four bits of one data byte and the leading four bits of the next data byte (or some other combination). Therefore, an unambiguous method of recovering byte boundaries is required.

The most straightforward method of recovering byte boundaries is through synchronization. Along with the data, a synchronization stream, which consists of a predetermined, unambiguous sequence of bytes, is sent. Immediately after detecting the sync stream, the receiver knows that the next received bit constitutes the leading bit of a data byte, and interprets the data accordingly.

The synchronization process used in this project consists three sync bytes, followed by 100 data bytes, repeated indefinitely. To permit synchronization, the value 0xff (255) is not allowed as legitimate data; it appears only in the sync stream. Data resolution is thereby reduced from 255 bytes to 254, a negligible difference. The synchronization stream consists of the following three bytes, transmitted LSB first: 0xff ff 00. As far as we can tell, under Manchester encoding, this sync byte can't be "faked" by legitimate data.

Synchronization works as follows. Initially, the base station is in a non-synced state. Once the clock is recovered, the received bits are continuously shifted through a three-byte software register. After each new bit is shifted in, the register is compared against the known sync stream (Figure 11). As soon as they match, sync is found, and the base station begins to receive data. Once 100 data bytes have been received, the receiver is forced out of sync, and waits to receive the next sync stream. Forcing the receiver out of sync increases the transmission's robustness; if for some reason, the receiver gets unwittingly desynchronized while receiving data, the problem is corrected at the next sync stream, and no more than 100 data bytes (less than one-half second of data) are lost.

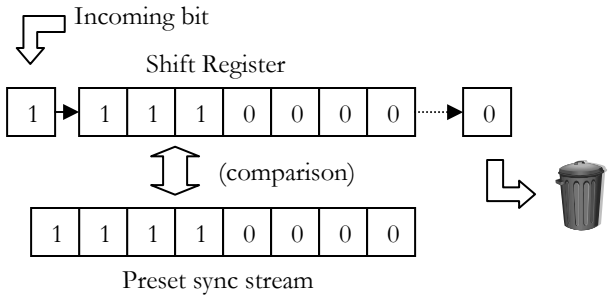

Figure 11. 8-bit example of how the incoming data register is compared with a preset sync stream. The actual sync stream utilized in this project is 24-bits.

We found that, in practice, the received signal's pulse widths often deviate from their expected values, likely due to imperfect signal recovery between the base station's antenna and the U2270B. Occasionally, the pulse widths can vary by as much as 50%. Although this undesired pulse width modulation tends not to confuse long and short pulses, it does leave the data integrity in question. In particular, when the antennas are moved far apart, close to their maximum range, received data consists of garbage with pulse widths that are much narrower than usual.

To combat this, measured pulse widths are compared to expected values using a hardware timer. If the pulse widths are far enough from the expected width of a long pulse (513 μs) or a short pulse (256 μs), the current byte is labeled as junk. If three junk bytes are received in a row, which may signal a transmission problem, the receiver is forced out of sync.

To ensure consistent timing at the PC, data is sent to the PC from the base station at the same rate that it is received. Legitimate data bytes are transmitted unadulterated as soon as they are received. A value of 0xff is transmitted whenever a junk byte is received. While the base station is syncing, it transmits 0xff bytes at the data rate (about 240 bytes per second), controlled by a hardware timer. This constant data rate ensures that timing remains consistent, without any complicated procedures on the PC. Additionally, the Matlab GUI keeps track of all received non-data bytes (values of 0xff), and can plot them at their received times. That way, transmission problems can be clearly determined and resolved.

The robustness of data transmission was tested in several ways. First, the transmitting and receiving antennas were placed close to one another and data was transmitted. Then, transmission was started, and the transponder's antenna was periodically moved out of range of the base station (Figure 12). It was anticipated that the received data would show an absence of data points when the antennas were out of range, and rapidly reacquire the signal when they were brought back together. As expected, no data integrity problems or synchronization errors manifested themselves. We therefore conclude that data transmission is quite robust, and should easily meet the experimenter's needs.

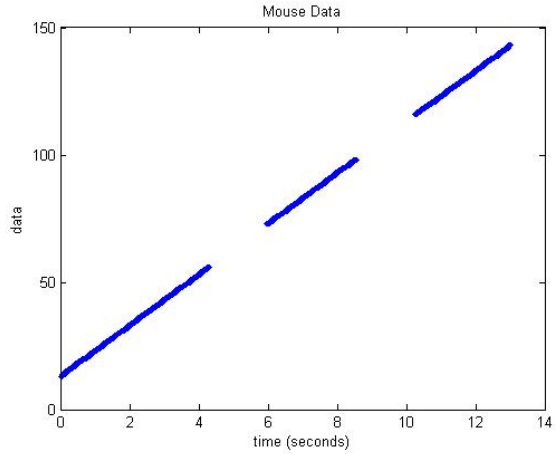

Figure 12. Stair-stepping data received at base station while transponder antenna was moved. The two gaps in the data are from when the antennas were out of range. Note the lack of corruption in the legitimate data.

VI. Conclusions and Future Work

A great amount of work has been accomplished in the pursuit of this mouse implanted RFID system. Data transmission works well, and the maximum antenna range, which is around one-half of a foot, is acceptable. The system is quite usable, and requires minimal knowledge of its inner workings.

Nevertheless, in their current forms, neither the transponder, nor the base station, is ready for actual experimental use. Currently, the base station makes use of a simple circuit that does not provide maximal transmission distance. A more complicated circuit is suggested by Atmel that increases transmission distance, but contains many more circuit elements (Figure 13). This circuit was constructed, but not fully debugged. To increase the final system's transmission distance, this more completed base station circuit should be perfected.

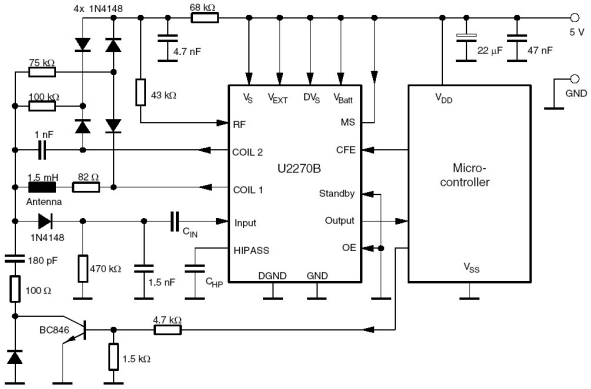

Figure 13. More complex base station circuit for increased transmission distance. Image courtesy [2].

The base station was developed with supporting circuitry on a relatively bulky protoboard, and the Mega32 was used within an STK 500 development environment. Before the base station can be used in actual experiments, all of its components should be soldered to a single solder board or printed circuit board, and packaged within a somewhat sturdy container. This procedure, though potentially time-consuming, requires little additional research and would be simple to perform in the future.

The transponder also requires additional work. During experimentation, the Tiny13V and LM61 temperature sensor were both powered from an STK 500 development board. The final design calls for both components to be powered from the U3280M, using RF-supplied current from the base station. We have reason to believe that both components would function well if powered from the transponder. The U3280M is capable of supplying 15mA of current, and in their current configurations, the Tiny13V and LM61 together consume only 2.6 mA. However, time constraints precluded testing this theory.

Separating the transponder from the STK 500 entirely is a major priority for future work.

Additionally, we had a lot of difficulty in miniaturizing the transponder's antenna. A reasonable transponder antenna should have a diameter of less than one inch (preferably, less than half an inch) for implantation in a mouse. The antenna we used for testing was built without this requirement in mind, and had a diameter of 3.5 inches, with an inductance of about 737 μH. Because the communication range is determined by the inductance of the transponder's coil, the smaller-size coil needed to have a similar inductance to achieve comparable performance. Our attempts at winding a ferrite-cored 0.33"-diameter coil were unsuccessful. A 73-turn coil had an inductance of only 73.7 μH, one tenth of what we needed. Since inductance is proportional to the square of the number of turns, we estimated that we'd need about 3.2 times as many turns (for a total of 230) to achieve an inductance of 737 μH. With the diameter wire that we were using, this would have resulted in a rather large coil, so we decided not to undergo the time-consuming task of attempting its construction. In the future, very small diameter wire, and perhaps some sort of automatic-winding apparatus, could facilitate the construction of a miniature coil.

Finally, the transponder too needs to be soldered to a small board and packaged. Again, this process would require negligible additional research, and could be accomplished relatively simply.

Though there is work yet to be done before the mouse-implanted RFID system is operational, we have accomplished a great deal this semester. We look forward to continued work on the project, and we expect that the day when the system has an opportunity to prove its mettle under actual experimental conditions is not far off.

VII. Cost Summary

One major advantage of this design is its very low cost. A commercial version, including base station, might retail for far more than the $25 that we estimate this system to cost (Table 1).

U3280M	$2.10
U2270B	$2.58
Tiny13V	$1.40
Mega32	$8.28
LM61	$1.04
Misc (estimated)	$10.00
Total	$25.40

Table 1. Cost summary.

These numbers represent Digikey (www.digikey.com) low volume prices as of December 14, 2004 (with the exception of the U3280M, which was not available low volume). "Misc" represents miscellaneous circuit components, such as resistors, capacitors, diodes, antenna wire, solderboards, et cetera.

VIII. Work Distribution

The mouse-implanted RFID system is currently being developed by Daniel Golden (the author) and Diana Rodriguez, under the guidance of Bruce Land at Cornell University. Diana started the project over the Summer of 2004, developed the base station and transponder circuits, wound antennas, coded the transponder, and performed general testing. Dan joined the project in Fall, 2004, wound antennas, developed the data encoding/decoding scheme, wrote the Matlab GUI, coded the base station, constructed an unsuccessful version of the more complex base station circuit, and performed general testing. Bruce initiated the project, and provided guidance and support throughout.

IX. References

[1] Atmel Corporation, "Tag Tuning," accessed December 11, 2004, online at http://www.atmel.com/dyn/resources/prod_documents/DOC2055.PDF

[2] Atmel Corporation, "U2270B Read/Write Base Station Datasheet," accessed December 11, 2004, online at http://www.atmel.com/dyn/resources/prod_documents/doc4684.pdf

[3] Atmel Corporation, "U3280M Transponder Interface for Microcontroller Datasheet," accessed December 11, 2004, online at http://www.atmel.com/dyn/resources/prod_documents/doc4688.pdf
[4] Atmel Corporation, "Electronic Immobilizers for the Automotive Industry," accessed December 11, 2004, online at http://www.atmel.com/dyn/resources/prod_documents/doc4661.pdf
[5] "Air Coil Calculation," accessed December 11, 2004, online at http://hem.passagen.se/communication/aircoil.htm