Funetics

Maureen Flintz, Edwin Ortiz, Meychele Chesley, and Daniel Falconer

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — This electrical and computer engineering project, dubbed Funetics, combines electrical and computer engineering learned aspects including audio file processing and RFID sensors. The project plays phonetic audio MP3 files. Which MP3 to play is determined by RFID cards placed on continuously active RFID sensors. Hand crafted code was written to pair RFID cards with phonetic sounds, read said cards with always active sensors, and play the sounds sequentially and repeatedly until the cards are removed.**

***Index terms* — Phonemes, RFID, speech, neural network, IPA**

I. INTRODUCTION

Funetics is a fun and innovative design built for many different types of users with communication ailments. Funetics acts as an Alternative and Augmentative Communication method that aims to supplement or replace speech. Funetics is essentially a box 6 slots in the top, where a user can insert a card that has a unique phenome displayed on it. By placing more cards and adding the phenomes together, the user will be able to create and hear correctly pronounced words.

We chose to have Funetics implement the International Phonetic Alphabet (IPA), the most commonly used alphabetic system of phonetic notation. The IPA contains over 40 different symbols representing each phonetic sound in the English language, but in order to simplify the concept, we selected specific key words for Funetics users to be able to choose from. These pre-selected words will be displayed on sample cards. The sample cards will have the word that the user can build spelled in English, a picture of what the word means and the IPA spelling of that word. This makes is possible for anyone to use our device and does not require the user to have any previous knowledge of any kind.

In order to make the interaction with the device fun for users a wireless RFID system is utilized. Passive RFID cards with known alphanumeric IDs are placed within range of six RFID sensors. Those ID numbers are processed by the microprocessor. The Atmel atmega328p microcontroller is at the core of the self-designed PCB board which also contains the majority of the system components. The ID numbers are associated with specific MP3 files which contain a variety of phonetic sounds as well as full words. The MP3 files are stored on an SD card. As the program continuously checks the RFID sensors for the presence of a card, the program will call the necessary files for playback. The files are sent to the DAC to be processed and then through the op amp before being output through headphones or speakers.

II. HARDWARE COMPONENTS

The primary hardware components of the system can be separated into a few different categories, each maintaining control over a major step of the overall process. The main components are the microprocessor, the wireless sensors, the audio chips, and the power supply. The primary components are integrated together with the main software component being the code on the Atmega328p. This section will be used to provide a basic introduction to each component.

A. Microcontroller

The core of the project is run by Atmel's atmega328 microcontroller. One of the major reasons for this choice was due to its use in the popular Arduino Uno platform and the associated resources. The 16MHz clock speed, 2 KB of RAM, and 32 KB of storage were all sufficient for the needs of our project. The Arduino IDE, which offered a user friendly, albeit simplified, environment that was used to program the atmega328. Most importantly, the atmeta328 offers 14 Digital I/O pins which is necessary for the projects number of RFID sensors[1].

B. RFID Sensors

The MFRC522 RFID sensor was chosen primarily for its easy integration with the atmega328. By default, the 522 works over a SPI communication with the atmega328 which worked for the needs of this project's design. The practical range of the integrated antenna is about 1 centimeter. This wasn't ideal, but the overall design of the project was modified to accommodate this distance. The 522 operates on a standard 13.56MHz frequency signal and can be used in conjunction with many standard passive RFID tags that come in various shapes and sizes.

C. *Digital-to-Analog Converter*

The MCP4921 is a 12-bit digital to analog converter with SPI interface. The Atmega328p uses the DAC_CS (chip select), DAC_CLK (data clock), DAC_DI (data), and DAC_LATCH (convert the digital to analog) pins to send the sample data over. The DAC also has a Vref input. This is the reference voltage that it uses to define the maximum analog value it can generate. There are two low-pass filters connected to the output of the chip. The first one is to prevent any digital noise from getting into the audio signal. The second low-pass filter connected to the output of the DAC and is for filtering out the square wave component in the recreated audio wave. While the noise is only 1/4096'ths of the signal (about 1.2mV) it's still noise and these two components filter out anything above 11kHz. The reason the filter cut-off frequency is 11kHz and not 22kHz is that if you sample at 22kHz you will only be able to reproduce frequencies at half that rate[3].

D. *Operational Amplifier*

The TL072 is a somewhat advanced JFET-input operational amplifier. It is an integrated circuit that incorporates well matched, high-voltage JFET and bipolar transistors. The TL072 has negligible input bias currents and is a low-cost part. It has low power consumption, which is ideal for our battery power design. Output short-circuit protection, low total harmonic distortion (0.003%), and high-input impedance are also included in the design.

E. *Quad Bus Buffer*

The SN74AHC125 is a level shifting IC that allows 5V and 3V components to work together. The chip will be powered with 3.3V and will allow 5V logic to be read on the input pins. This component is needed since the SD MMC reader runs on 3.3V while the Atmega328p runs on 5V.

F. *Power Supply*

For this project a 7.4V lithium ion battery with built in PCB protection to prevent over charging is used. The battery is connected to a NCP1117 voltage regulator with a fixed output of 5V. A second voltage regulator, the LP2985, is used. This regulator will take in the 5V and have a fixed output of 3.3V. The components powered by 5V are the ATmega328p, TL072P and the MCP4921. The RFID readers along with the 74125N and SD MMC reader are powered by the 3.3V supply.

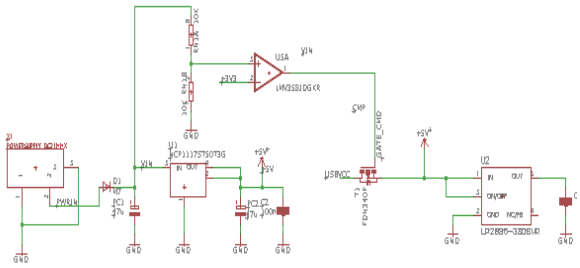


Fig. 1. Power supply schematic

G. *Audio Output System*

There are two options for audio output. The first option is to use the 3.5mm stereo jack which gives the most power output. The other available audio output option is to have the audio output to the PCB integrated speaker connection located next to the jack. If both the stereo jack and speaker connections have devices connected to them there are internal switches in the jack that, when the headphones are removed, will output the audio through the speaker.

III. SYSTEM DESIGN AND CONCEPTS

In order to better understand using this device and how it works, the following state machine diagram is helpful.

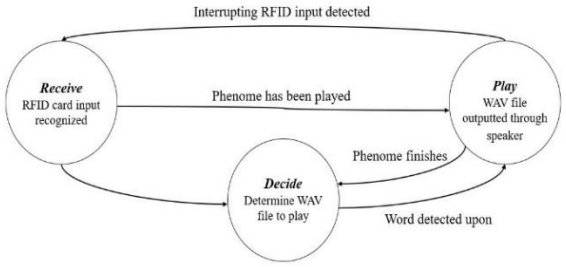


Fig. 2. Funetic audio state machine

As can be seen in the above diagram, the device system is cyclical. In other words, the system is continuously active in checking for new/existing RFID cards and playing the audio associated with those cards.

First the RFID sensors detect an input in the form of an RFID card placed within range. This can happen at any time. The MFRC522 RFID sensors are constantly refreshing to take input and check for any changes. Each

RFID card has an identification number pre-programmed from the factor. These identification numbers are used to associate the cards with specific phonetic sounds. It's possible, but highly unlikely, that a card could be missed. The read rate for the sensors is less than 1 second. RFID cards are read by the sensors continuously from left to right. The audio associated for that card will be played during the next cycle. If a card is placed down and picked back up quick enough it's possible for the system to not read that RFID card. However, that's not how the device is intended for use and it's not reasonable to assume a user will do this expecting it to work.

The next phase for the system to accomplish is to get the MP3 files stored on the SD card. This state is reached once all RFID sensors have read sequentially. As previously stated, every RFID card has a factory programmed alphanumeric identification. These IDs are then directly associated with the file names that correspond to the audio file of the phoneme represented on the card. A function then retrieves those the audio files needed from the SD card and places them in the same order as the cards are placed on the sensors. In the event a sensor is skipped and does not have a card on it, a function accounts for this and removes the blank space when the cards are read.

Finally, the audio is played. The MP3 data is retrieved from the SD, sent to the DAC via the atmega328p microcontroller, amplified, and then output through either the 3.5mm audio jack or a speaker (if connected to the PCB). The audio files are played in the same order as their RFID card counter parts are laid onto the device. An additional function will check the order of the cards and check them against known words is a premade library database. If a match can be made, an audio file of the word as a whole will be played after the phonetic sounds. Once all audio files have completed playing, the process will repeat with the latest read sequence of RFID cards.

IV. HARDWARE DETAIL

The major hardware components were outlined in the second section, Hardware Components, of this document. These same components will now be discussed in detail.

A. Atmega328

The ATmega328 is part of the Atmel AVR microcontroller family. It has a data-bus architecture and internal registers that are capable of handling 8 parallel data signals. The three types of memory that can be utilized by the ATmega328 are Flash (32KB), SRAM (2KB), and EEPROM (1KB)[1].

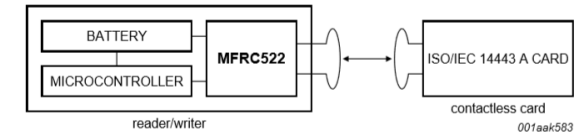
This MCU is available in a DIP-28 package, meaning it has 28 pins. These pins include power and I/O pins. Many of the pins are multifunctional this allows the same pin to be used in different modes based on how the user configures it. This reduces the need for a larger pin count since the MCU does not need a separate pin for every function. This allows a user's design to be more flexible since the I/O pins can provide multiple types of functionality[1].

TABLE I
ARDUINO ATMEGA SPECIFICATIONS

Clock Speed	16MHz
RAM	2KB
Storage	32 KB
Operating Voltage	1.8-5.5V
Minimum Current	42 mA
Digital I/O Pins	14
PWM Pins	6

B. RFID MFRC522

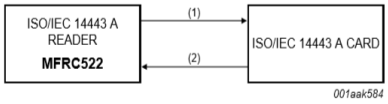
Radio-frequency Identification utilizes electromagnetic waves to identify objects that are RFID tagged. These cards act as identifiers for the object with the tag. Some implementations are passive, so they rely on energy collected from the interrogating, or reader, radio waves of nearby RFID readers in order to send information. An Active Reader Passive Tag (ARPT) system includes passive tags that are activated by the interrogating waves of the active reader. The range of passive tags, lack of a power supply, and the small amount of storage space are tradeoffs to be considered for the upsides of passive RFID tag systems. Passive tags are very low cost. The lack of a built-in power supply, the small amount of storage, and the simple fact that passive tags are sometimes meant to be disposable make for low manufacturing costs, which results in a low price point for buyers.



MFRC522 Read/Write mode

Fig. 3. MFRC522 Read/Write mode

The MFRC522 RFID sensor transmits the full 16-byte RFID data word immediately upon receiving it from the transponder. It is transmitted via ISO/IEC 14443 A/MIFARE at 10Mbit/s when using the SPI communication[2].



Reader to card 100 % ASK, Miller encoded, transfer speed 106 kBd to 848 kBd.

Card to reader subcarrier load modulation, Manchester encoded or BPSK, transfer speed 106 kBd to 848 kBd.

ISO/IEC 14443 A/MIFARE Read/Write mode communication diagram

Fig. 4. ISO/IEC 14443 A/MIFARE

V. HARDWARE DESIGN CONCEPT

With the hardware components detailed the overall high level design of the project can be described. In the following block diagram it can be seen how the major components are integrated as well as the I/O flow.

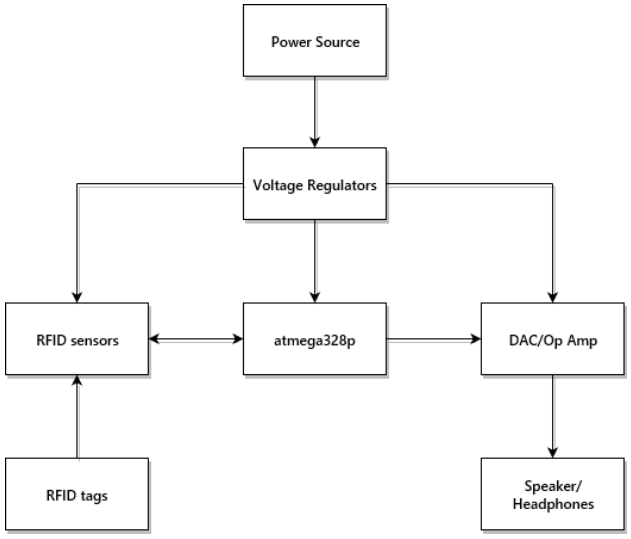


Fig. 4. Hardware block diagram showing major components and I/O flow.

VI. PRIMARY SOFTWARE CONFIGURATIONS

The overall software configurations for this project can be split into two major components. It is composed of the embedded system, the main functionality, and our neural network voice model.

A. Embedded System

The heart and soul of our project. The embedded system is developed in C which controls the entire system. The general procedure follows a standard device layout that consists of a setup portion and a continuous loop portion.

Initially, our setup begins with turning on all the RFID sensors and creating a unique instance class for each one. Then, we initialize the SD card to allow fast and direct file access by using FAT partitions. The SPI configuration is set to communicate at 20Mhz so that the SD card can send the data stream as fast as possible. This is very important as a lower frequency can cause audio slowdown or distortion. Once SD communication is established, the system will announce a starting up audio to indicate we have successfully turned on with no problems. The last component of our setup is now to run a separate timer that will keep track of how often to attempt producing audio of a completed word.

The main bulk of our functionality now lies in the continuous loop. Due to our unique instance classes for the sensors, we can easily turn on one at a time so no sensor is ever interfering with another. The time delay between turning one on and off is so small it is negligible while providing the effect of it working simultaneously. As each sensor is turned on, it will check if any phonemes are present. If one is present, it will enunciate how to say that phoneme while storing it to assembly with the other presented phonemes.

After reaching the time duration we previously assigned in the setup for our separate timer, all the currently presented phonemes will be strung together, removing empty spaces, and that string is compared to our dictionary of words in our SD card. If it is a real English word, the system will pronounce it.

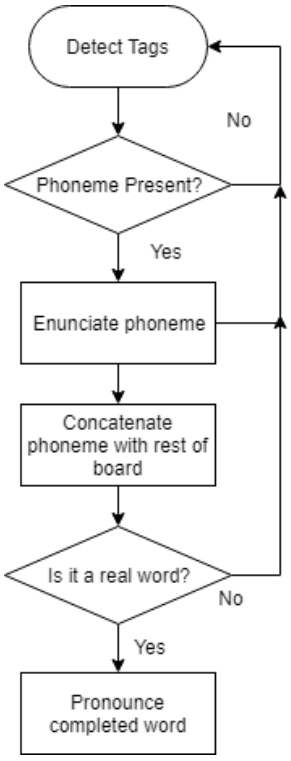


Fig. 5. Flow chart representing the embedded system process flow.

B. Neural Network Model

The voice you hear after the time duration is reached from the separate timer, is a computer-generated voice. The purpose of using a computer voice was to create the mass audio dictionary to contain as many words as possible pronounced by the same "voice." With an audio dictionary on board of the embedded system, it allows portability, offline "text to speech", rapid and instantaneous access times.

We created this voice through the use of Python. Using Tensorflow, we can input a sample audio file with what it is supposed to be and have the network learn. With the use of a trained neural network model, we were able to input any given string and output a corresponding audio file. Upon creating our audio files, we had to configure the specifications to fit our designs. All audio is converted to 8-bit, 11 kHz or 22 kHz, mono wav files.

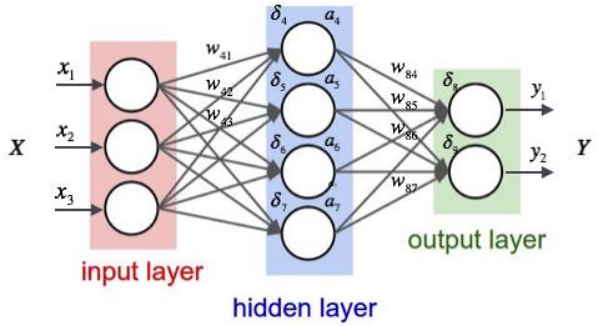


Fig 6. Neural network process. The diagram represents the layered implementation of training a convolutional neural network.

The audio is then played by creating a Wave class instance of the file using a FatReader to extract the wave file's metadata. After filling the buffers, the samples are played back.

C. Software Details

Assignment of phonemes are unique to specific RFID tags that allow us to differentiate between sounds. Typically, each sensor only detects new tags. This is problematic as our tags will be sitting there so they will no longer be new as it cycles through detection again. This can be overlooked by constantly overriding the sensors to a ready state and clearing the buffer request array at the end of each detection cycle. The UID of a tag is converted into a single byte char phoneme representation that gets concatenated with the file extension type. Regardless of empty spaces, phonemes get read left to right by keeping track of two arrays with two pointers. We will only concatenate strings that are not null char. This allows a phoneme to be placed at the very first slot and the very last slot and still return a two-phoneme word.

VII. DEVICE HOUSING AND DIMENSIONS

A. Housing Dimensions

The primary prototype of Funetics is made from wood. The box is approximately 34 inches x 8 inches x 6 inches. This large size is required to accommodate the RFID sensors and RFID cards. There is about 2 inches between each sensor and card placement area. This is required for both the placement of the cards to not feel cramped as well as provide enough space so that the RFID cards aren't close enough to be picked up by adjacent RFID sensors. Given this large size there's ample room inside to contain the PCB, the breakout board for the RFID sensors, and the RFID sensors themselves.

B. Wiring Electronics

The electronics for the device will be wired together inside of the housing using 22-gauge electrical wire with different colors to represent different connections. Each of the six RFID readers has seven connections that need to be wired to a perfboard to accommodate multiple connections to the PCB. The majority of the SPI communication lines can share a connection, so it was decided that a perfboard was the most secure way to make the connection to the PCB. The PCB and the perfboard are secured to one of the walls in the middle of the housing this will allow for shorter connections and more organized wiring.

VIII. BOARD DESIGN

The system design, with exception of the RFID sensors, is implemented on a 2-layer printed circuit board. The PCB schematic was designed using the EagleCAD PCB design software. After testing all the applicable components using a combination of breakout boards and a breadboard, the design was then reproduced virtually and sent off to be made. OSHPark was the company that fabricated the PCB.

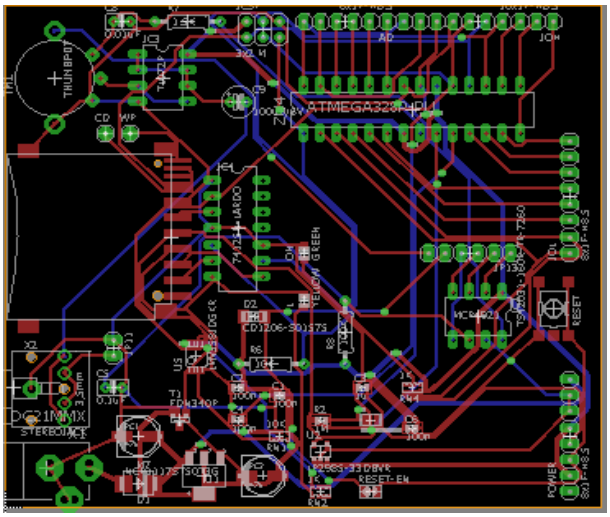


Fig. 8. Printed circuit board layout

IX. DEVICE SAFETY REQUIREMENTS

For a device that is meant to be used and interacted with by other people it's paramount that Funetics be safe to use by all. The word "Fun" is in the title after. In order for Funetics to be fun it needs to be safe. As a device that's meant to be a child educational tool one must be aware of any possible dangers. The basic function of Funetics is to playback audio for others to listen to. There are physical limitations to the intensity of sound that the human ear is capable of withstanding without damage and that level must be kept in mind during development. For the human ear, lengthy exposure to sound levels at 85 dB or above can cause damage. This device is meant to be used in a teaching environment, so if it is possible for Funetics to get louder than 85 dB a warning will be included with the device.

X. ADMINISTRATIVE

A. Group Dynamics

The project tasks were distributed between the two CpEs, Edwin and Meychele, and two EEs, Maureen and Daniel. The two CpE students focused on choosing and implementing a microcontroller and designing software for the project. The two EE students focused on choosing and implementing the hardware aspects of the project including the sensors, DAC/Op Amp, and power supply, as well as the task of designing the PCB.

B. Budget and Finances

The project is not sponsored and is funded by the group. The cost has been split between the four group members. The cost of this project in the final stages was approximately $450. Some parts we were able to receive at no cost. Keeping the cost as low as possible is important since the project was not sponsored.

XI. FUTURE PLANS FOR FUNETICS

Due to our time and cost restraints, our current project is scaled down to a limited availability of phonemes. The information publicly available to train a neural network to pronounce phonemes is so sparse that creating it would require many samples and time to produce quality up to par as its full word variant. Additionally, we have only included a small set of phonemes in which you can attempt to construct words with. This greatly reduces the amount of variety a user could try and create. Ideally beyond the scope of senior design, it would be beneficial to continue implementing all the phonemes and create a full set dictionary to allow usage of full speech. Futhermore, Funetics could be implemented to incorporate the same functionality but in different languages since the IPA should be able to use to construct any combination of sound therefor any word in any language. It would just be a matter of creating the IPA transcriptions for the specific language.

XII. DIFFICULTIES

A. Hardware

From start to finish, stable connection between components was a constant slow down giving us a variety of issues. Whether we used a perfboard, a bread board, or any method to connect our wires and components, we never had a connection where we could turn it on and confidently say it will work. If one wire was loose or not fully connected, then the entire set up would be affected. It was obviously most noticeable that there was a wire connectivity problem when the software was running slower than expected. We would dissemble the whole thing and put it all back together because trying to pin point which wire was causing an issue would have taken the same time, if not longer. The circuit may not have been working for us initially, but then dissembling the entire circuit and putting it back together the same exact way would allow it to work again.

Additionally, the SD card was tremendously reliant on a stable connection. The component to read our SD card was the most sensitive part of our project. We would run into errors with initialization or we would be unable to read files out of the SD cards often.

Getting up to 6 RFID sensors to all work properly was challenging as well. There were numerous times where we were unable to get all 6 working and considered downscaling to a lower number of sensors. We theorized the number of sensors might have been causing an overflow of too much communication between the slave lines and microprocessor.

Similarity to the sensors, the SPI line being overloaded was causing a slowdown. Pulling out data from the SD card was going slower than desired due to so many components and interactions going on and constant rapid calculations.

Both the sensor and SD speed/communication problem was solved by adjusting the frequency and clock rates.

B. Software

Reading the SD card and collecting the input from the sensors were so slow that it could take nearly 30 seconds just to read and hear the pronunciation of one phoneme. In the software end, we had to configure the frequency at which the clock and SPI were going on to make sure we could produce speeds at an optimal rate. The SPI communication is set to run at maximum speed at 20MHz, while the clock runs at a maximum of 16MHz. Despite this speed ups, the audio still gave us problem on playback. To compensate for this delay, we had to reduce the frequency at which audio played. Originally, we would have liked to keep the highest bit rate and frequency as possible to produce the highest quality output for users to hear. But the best quality we could produce while keeping the clearest crisp pronunciation was to set the configurations to 8 bit and 11kHz for audio that is played in correlation to phonemes and string construction. When audio is played during the set-up portion before entering the main loop, audio can be played up to 16kHz.

XIII. CONCLUSION

We designed Funetics to be a hands-on interactive learning device. This device should provide the users with a fun way to learn the correct phonetic pronunciation of English words. Not only has Funetics challenged us technically, we were also challenged creatively, building something that can help others in a new and unique way. This project has been a great way to apply and expand our knowledge.

BIOGRAPHIES

MAUREEN FLINTZ is a 28-year old Electrical Engineering student. She is currently employed as a circuit board technician. She wants to pursue a career in test engineering.

Daniel Falconer is a 27-year-old Electrical Engineering student. He starts an internship at an engineering consulting agency this summer and dreams of one day becoming a full-time consumer electronics journalist.

Edwin Ortiz is a fourth year student who will obtain his Bachelor of Science degree in Computer Engineering at University of Central Florida at the end of the semester, Spring 2018. After years of experience working for many companies related to software engineering and development he hopes to one day start his own business improving quality of life through innovation and invention.

Meychele Chesley is a senior in Computer Engineering at the University of Central Florida. She has worked at a circuit board manufacturing company for some time and wants to use her hardware background to create innovative software solutions in the future.

ACKNOWLEDGEMENTS

The team would like to kindly thank the following people for their assistance, guidance, and support throughout our time at UCF and during Senior Design 1 and Senior Design 2; Dr. Samuel Richie, Dr. Lei Wei, Dr. Reza Abdolvand, Dr. Nazanin Rahnavard, Dr. Vikram Kapoor, William Shaw and Ruben Leon.

REFERENCES

[1]Y. Tawil, "Understanding Arduino UNO Hardware Design", *Allaboutcircuits.com*, 2016. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/understanding-arduino-uno-hardware-design/. [Accessed: 27- Oct- 2017].

[2] "MFRC522 Datasheet, PDF - Alldatasheet", *Alldatasheet.com*, 2017. [Online]. Available: http://alldatasheet.com/view.jsp?Searchword=MFRC522. [Accessed: 12- Nov- 2017].

[3] "12-Bit DAC with SPI™ Interface", *Ww1.microchip.com*, 2007. [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/21897b.pdf. [Accessed: 12- Nov- 2017].

[4] " Quadruple bus buffers with 3-state outputs", [Online]. Available: http://www.ti.com/lit/ds/sdls044a/sdls044a.pdf. [Accessed: 12-Jan-2017]