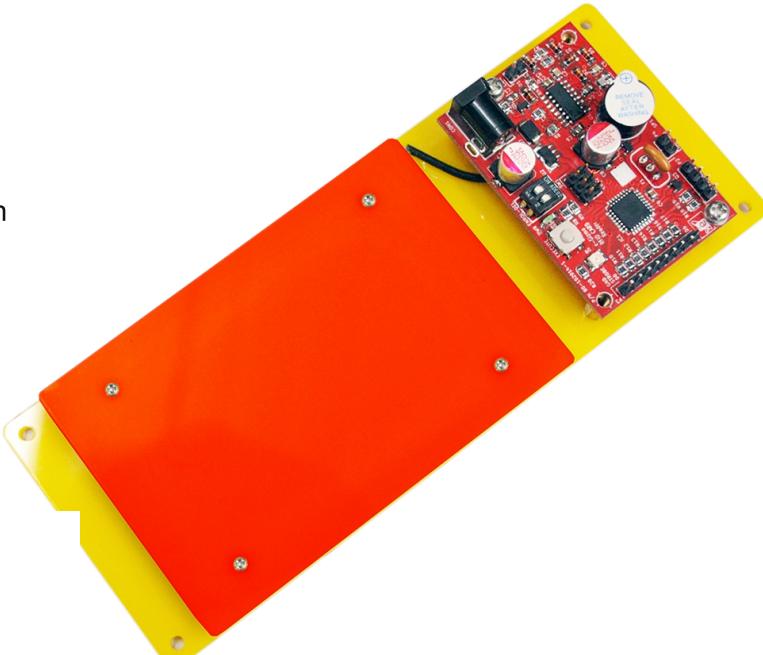# RFID Card Reader

Technical Manual Rev 2r0

The e-Gizmo The RFID card reader can store and authorize (enrol) up to 100 cards without the need for additional hardware (i.e. PC). Not all 125khz card are encoded in the 40-bit format recognized by this RFID card reader. To prevent compatibility issues, use only with qualified RFID cards and tags sold at e-Gizmo.

FEATURES:

- 100 cards storage stand alone operation
- Compatible with 125kHz
  40-bit encoded cards
- Serial UART port
- 7-bit Parallel port
- Lock Release output
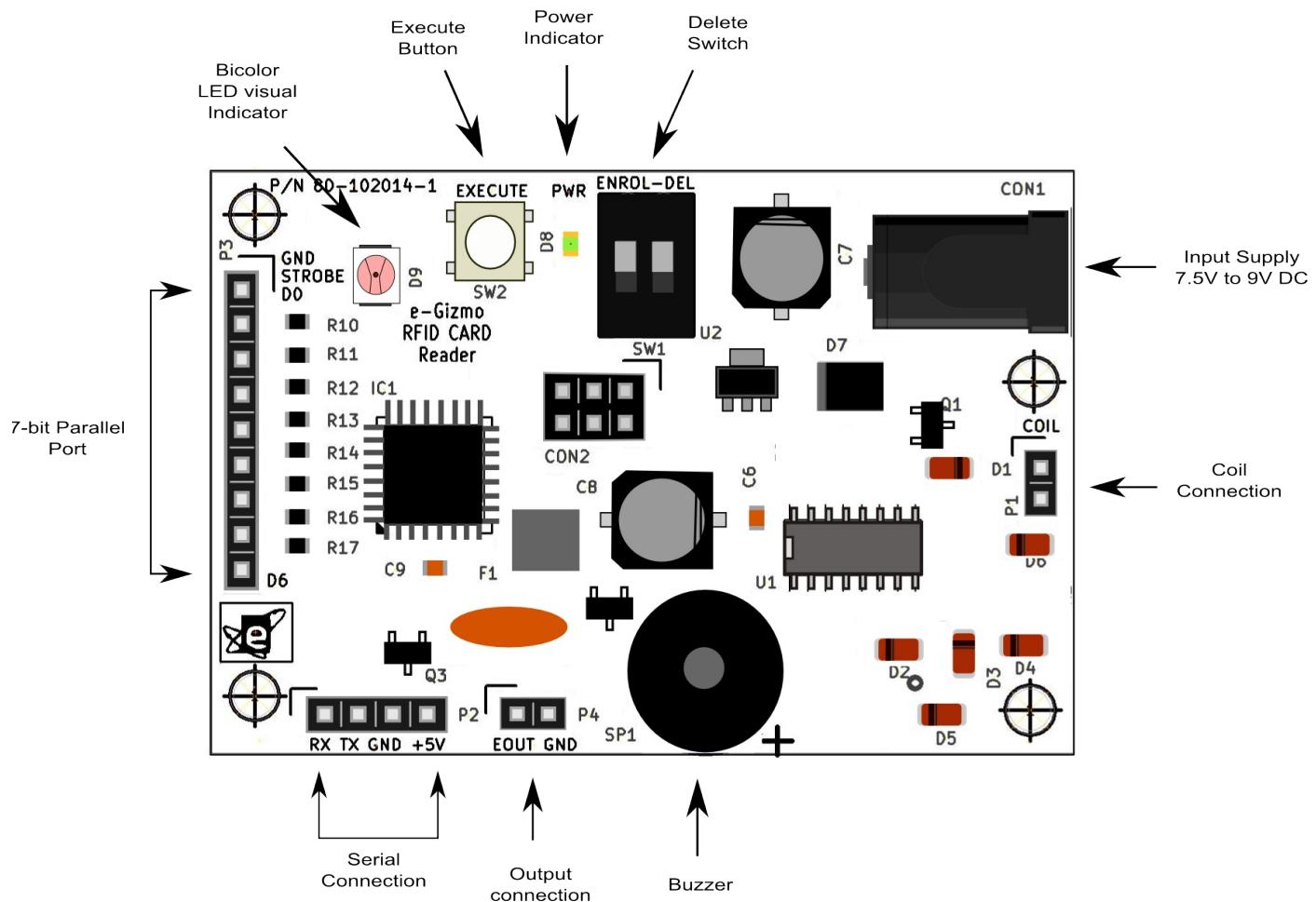- Bicolor LED visual indicator

RFID Card (125kHz)
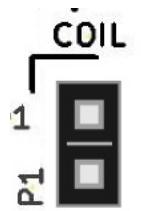
RFID Tags (125kHz)

GENERAL SPECIFICATION:

- **Supply Input:** 7.5V to 9V DC

- **On board IC:** ATMEL ATmega168

- **PCB Dimension:** 46 mm x 64 mm

Figure 1. Major parts presentation of e-Gizmo RFID Card Reader

Table 1. *P1* connections and descriptions

PIN Name	Descriptions
COIL	Pin Connection of coil

Figure 2. P1 Illustration

Serial UART Port

Allows access to advance functions and for external card readings, processing, and storage functions. Storage capacity is then limited only by the external hardware.

Table 2. *P2* connections and descriptions

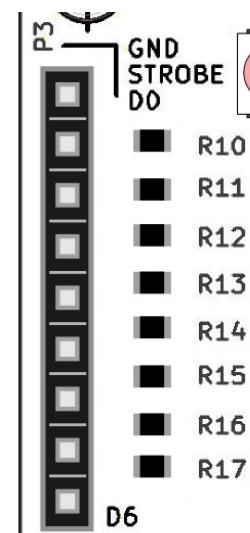| PIN Name	Descriptions
RX	Receiver pin connection
TX	Transmit pin connection
GND	Ground
+5V	Input Supply

Figure 3. *P2 Illustration*

7-bit Parallel Port

For users still wanting to do things using parallel data source. The 7-bit port outputs the corresponding memory address of an enrolled (registered) card. (See Figure 4)

Table 3. *P3* connections and descriptions
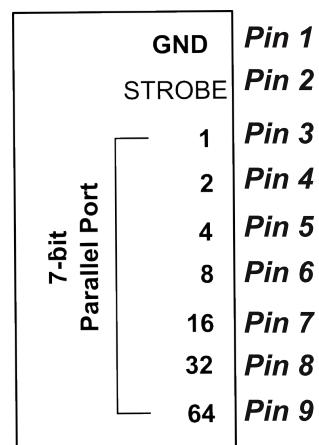
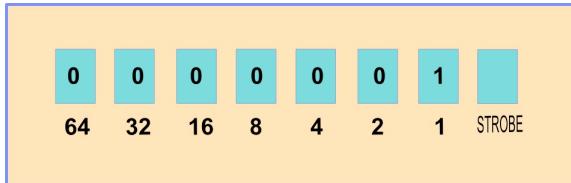| PIN No.	Descriptions
1	Ground
2	STROBE
3	D0 (Bit 1)
4	D1 (Bit 2)
5	D2 (Bit 3)
6	D3 (Bit 4)
7	D4 (Bit 5)
8	D5 (Bit 6)
9	D6 (Bit 7)

Figure 4. *P3 Illustration*

7-bit Parallel Port with Strobe

The 7-bit parallel port outputs and retain the memory address of the last swiped (enrolled)card. A normally high strobe output momentarily switch to low logic each time a new (enrolled) car is swiped and detected.

e-Gizmo RFID Card Reader

0	0	0	0	0	0	1	
64	32	16	8	4	2	1	STROBE

Example :

Address: 001 is ON High state
You can put a LED from GND to the
D0. To recognize the number of a user.

Note: STROBE is normally HIGH if there is
No Listing cards on memory.

Lock Release output

This open collector output momentarily
switch ON each time an enrolled card is
detected. This can be used to drive a relay
driver that in turns activates to release a lock.
Alternately, it can be used to drive an MCU
input to indicate a card had just been read and
validated.

Table 3. *P4* connections and descriptions

PIN No.	Descriptions
1	EOUT connection
2	Ground

Figure 5. *P4 Illustration*
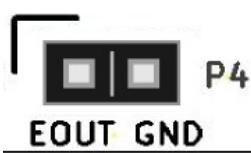
Important Notes:

Lock Release Output

1. Open collector NPN output. Compatible with
any digital input with 3V to 12V logic.

2. Interfacing to an MCU. Pull up resistor to Vcc
is REQUIRED. If used with a gizDuino/arduino,
the internal pull up resistor can be enabled
instead so that external pull up will no longer be
required. This applies as well to other MCUs
with internal pull up.
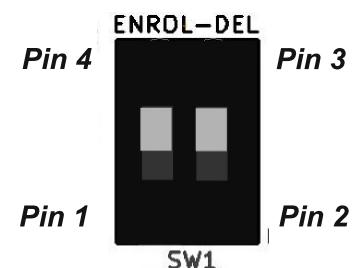
Enroll & Delete Switch

This will be the selection for Enroll or
Deleting the Card. If you want to enroll a card
just switch it form Pin4 - Pin 1. Then for deleting
all the cards you need to switch it from Pin 3 -
Pin 2, first. Before you send the corresponding
commands. Later you will learn how to do
delete it.

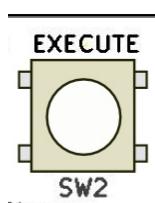**Table 4**. *SW1* connections and descriptions

PIN No.	Descriptions
1	STOR (Store)
2	DEL (Delete)
3	GND (Ground)
4	GND (Ground)

Figure 6. *SW1 Illustration*

Execute Button Switch

The Execute Button Switch is important **to ENROLL** specially if you want to enroll a card or multiple cards. Just **Press and Hold the SW2 while tap the card near to the coil**. Once you heard a beep sounds it means that your card is enrolled/ stored to the memory.

Figure 7. *SW2 Illustration*

Buzzer

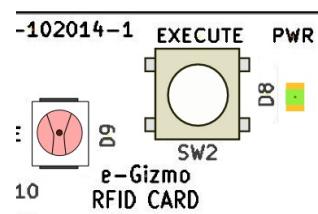A short beep will sound each time a card is read.
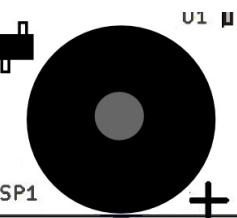

Figure 8. *SP1 Illustration*

LEDs indicator

Bicolor LED visual indicator (D9)

Provides visual indicator to the user if the card is read and validated. Unenrolled card will cause the LED to flash RED. Enrolled card will flash it GREEN.
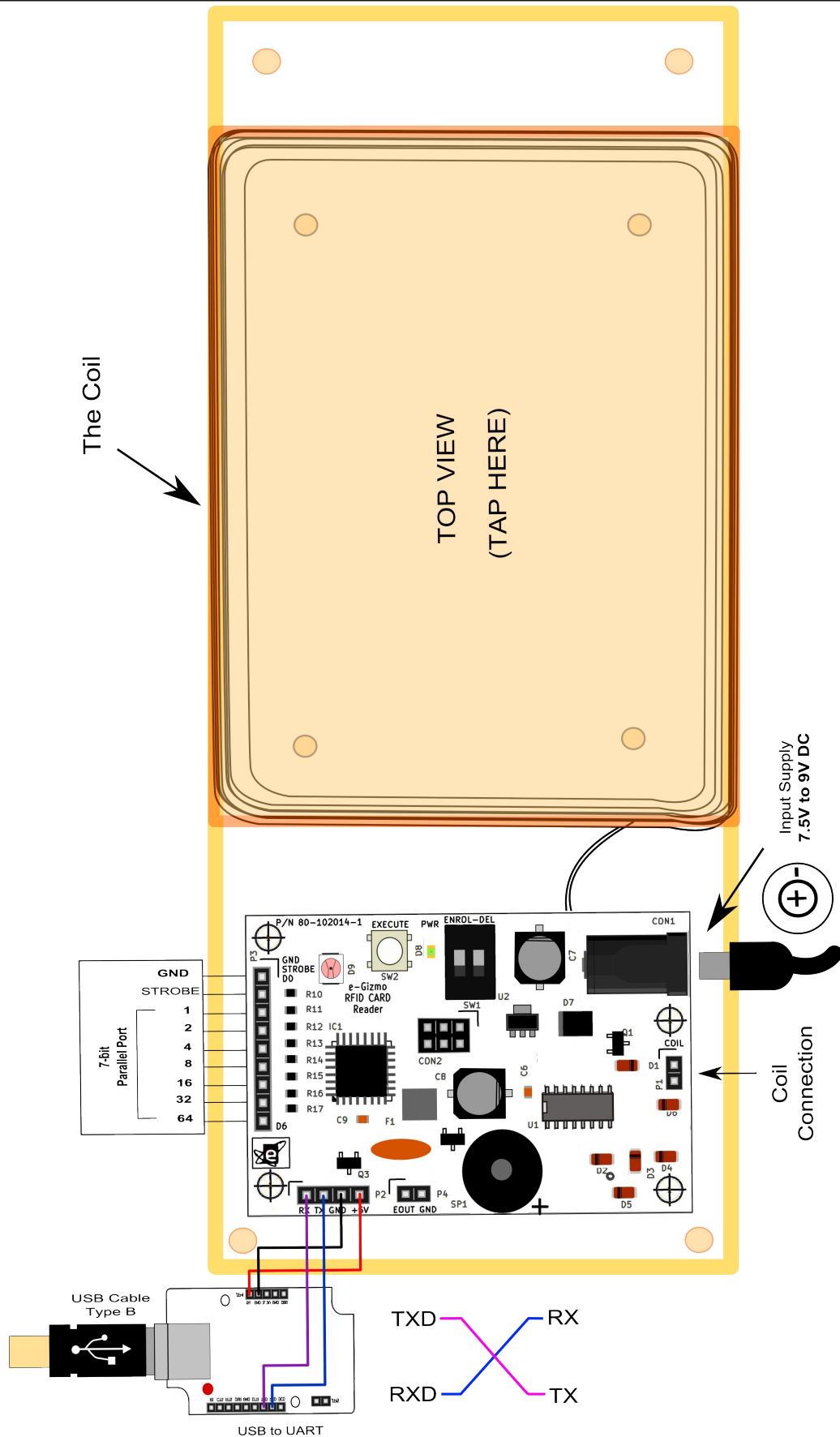
Power LED indicator (D8)

Figure 9. *D8 and D9 Illustration*

The Coil

TOP VIEW
(TAP HERE)

Input Supply
7.5V to 9V DC

Coil
Connection

GND
STROBE

7-bit
Parallel Port

1
2
4
8
16
32
64

USB Cable
Type B

USB to UART

TXD ——— RX
RXD ——— TX

Figure 10. Major parts presentation of e-Gizmo RFID Card Reader

e-Gizmo RFID Card Reader

Open Terminal v1.9b by Br@y++.

SETTING-UP

• *Select COM Port.*

COMMUNICATIONS MANUAL

• *Baud Rate:*   *9600*
• *Data:*   *8 Bit*
• *Parity:*   *NONE*
• *Handshake:*   *NONE*

Summary of Functions

vn	*- verbose ON/OFF:*
V	*- Firmware Version*
S	*- List card in Memory*
I	*- Initialize EEPROM*
Tn	*- Fine Tune ON/OFF*
Xnn	*- delete card*
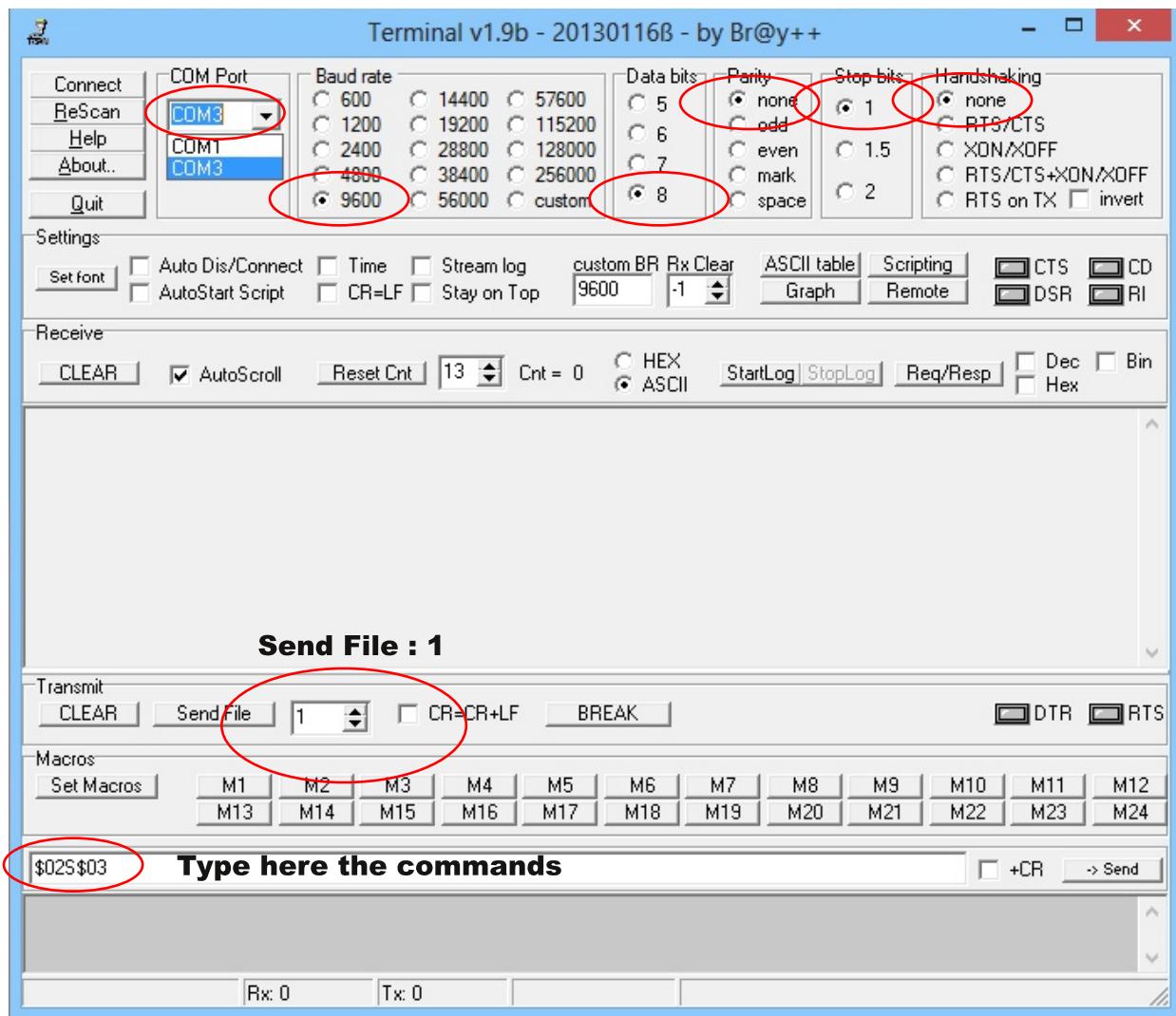
Figure 11. Setting Up Terminal Communications.

e-Gizmo RFID Card Reader

Communications Format

Every packet of data transmission are wrapped inside an [STX] and [ETX] marker.

[STX] – Start of transmission marker, ASCII value = 0x02
[ETX] – End of transmission marker, ASCII value = 0x03

The first character after the [STX] marker is a single character function specifier. Each transmission may contain just a function specifier only, or may contain a series of data in addition to the function specifier. End of transmission is signaled by the [ETX] marker.

[STX] and [ETX] are data packet markers and should not be transmitted as literal string. They should be send in their ASCII representation. The correct way of transmitting the [STX] and [ETX] markers are as shown in the following example:

Example 1: Show firmware Version

Transmission Format: **Format: [STX]V[ETX]**

This should be transmitted in their ASCII code representation as shown in the following table:

Symbol	STX	V	ETX
Hex	0x02	0x56	0x03

Visual Basic:

Correct:
' correct way to send [STX] & ETX marker
Serial1.print(chr(2)+"V"+chr(3))

Wrong:

Serial1.print("[STX]V[ETX]")   'WRONG!

Arduino:

Correct:

Serial.write(0x02);   // correct way to send [STX]
Serial.print("V");   // V
Serial.write(0x03);   // [ETX] marker

Wrong:

Serial.print("[STX]V[ETX]");   // WRONG!

Alternately, you can use the C/Arduino "\" operator to send the ASCII code of STX and ETX, together with the function and data:

Serial.print("\002V\003");   // "\002" = STX, "\003" = ETX

Notice that in the example, only the STX and ETX marker need to be manually converted to their ASCII code, for the simple reason that they have no equivalent printable characters. The three line implementation (long format)may make your program longer, but is more human readable. Hence, for clarity, all example codes given are shown in the long format. We leave it up to you if you want to convert and code it in short format.

READ CARD OUTPUT DATA FORMAT IN VERBOSE MODE "0"

Serial data output by default are presented in human readable format. To interface with a controller, it is much more convenient to use the CSV format, which can be activated by turning the verbose mode to off (see Function Description 1.0)

To turn of verbose mode, send
        [STX]v0[ETX]

With verbose mode off, serial read is now reported by the reader in the following format:

        [STX]e,aaa,ttt,nnnnnnnnn[ETX]
        where:
                e - enrolled? 1=enrolled, 0=not enrolled
                aaa - Internal memory location where card is enrolled, "XXX" if not enrolled
                ttt - card type
                nnnnnnnnn - card number

        Example:

        [STX]0,XXX,126,0009398762[ETX]
                e=0 - card not enrolled
                aaa= XXX - not in memory
                ttt= 126 - card type
                nnnnnnnnnb= 0009398762 - card number

©2014 by e-Gizmo Mechatronix Central   e-Gizmo RFID Card Reader

FUNCTION DESCRIPTION

1. **v** – *Verbose mode*

Verbose mode in ON state presents information in readable format. RFID replies are terminated with [carriage return] and are not wrapped with [STX][ETX] markers. This allows users to interact with the RFID conveniently.

Verbose mode in OFF state will generally suppress non essential messages and will present swiped card data in CSV format.

> ***Format: [STX]vn[ETX]***

> Where: **n** = Mode setting 0-1
> **0** - Turn OFF verbose mode (default)
> **1** - Turn ON verbose mode

2. **V** - *Display Firmware Version*

Returns a four digit Firmware Revision stamp.

> ***Format:[STX]V[ETX]***

Note: Reply always wrapped with [STX][ETX] even with verbose ON.

3. **S** - *Show all enrolled cards.*

Will show a listing of all cards enrolled in memory. Listing is always in readable format and each item is terminated with [Carriage Return].

4. **ue-Gizmo** - *Unlock EEPROM*

This will enable ERASE EEPROM function to work. Wait at least 100ms after the unlock function is invoked before issuing the ERASE EEPROM command.

> ***Format:[STX]ue-Gizmo[ETX]***

5. **I** - *ERASE EEPROM*

Intialize EEPROM to its blank state.
Caution: All cards in registered in memory will be erased.

` ***Format:[STX]I[ETX]***

Note: Unlock EEPROM first as described in 4. Prior unlocking minimize the chance of accidentally erasing all cards enrolled in memory.
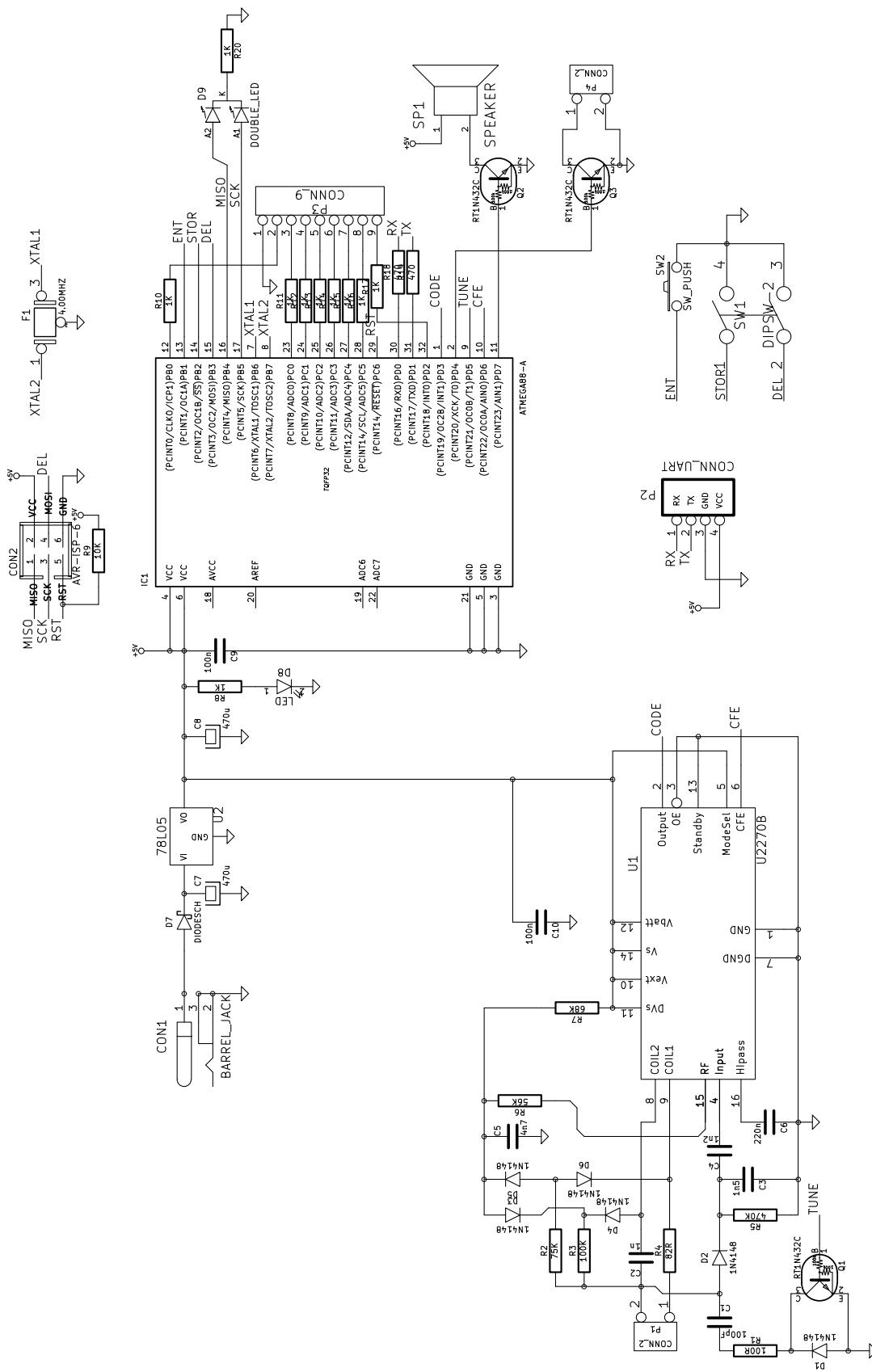
6. **X** - Delete a card

Allows deletion of enrolled card even when the card is not available.

> ***Format: [STX]Xnn[ETX]***

> Where: **nn** = Card address 0-99

nnn is the address assigned by the RFID reader during the time the card is enrolled. Use the S command to find the address of the card you wish to delete using this function.
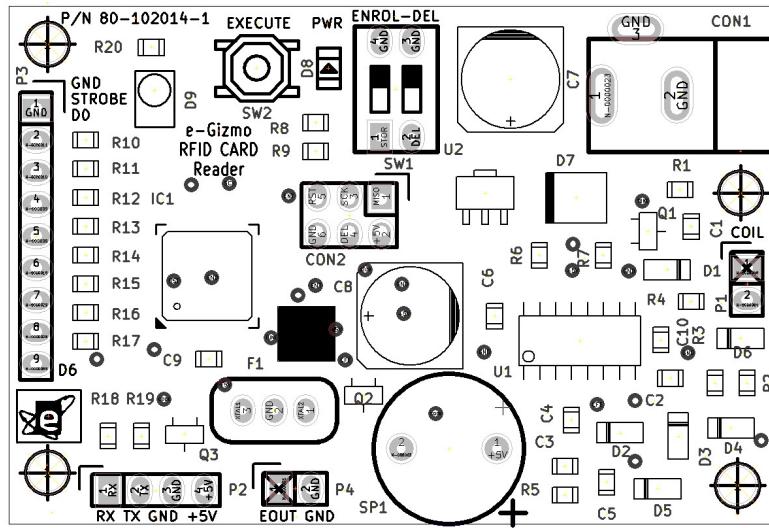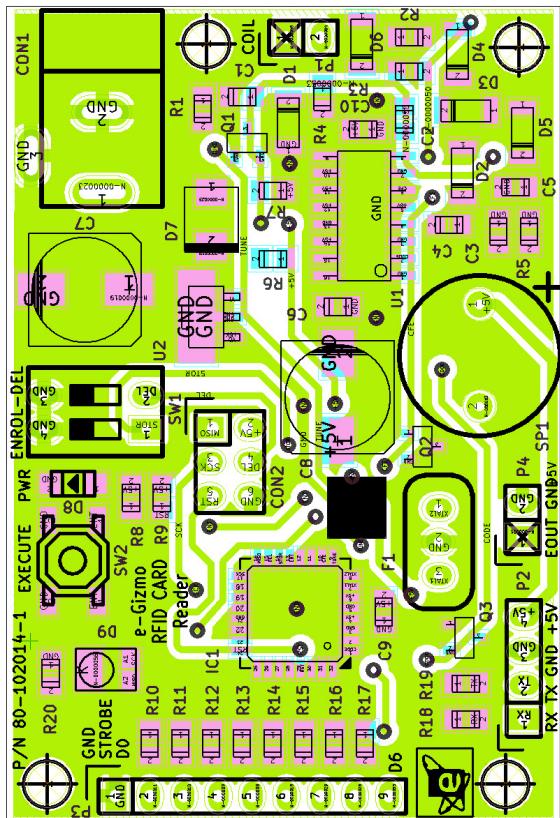
Figure 12. Schematic Diagram of e-Gizmo RFID Card Reader

Figure 13. *Parts Placement*
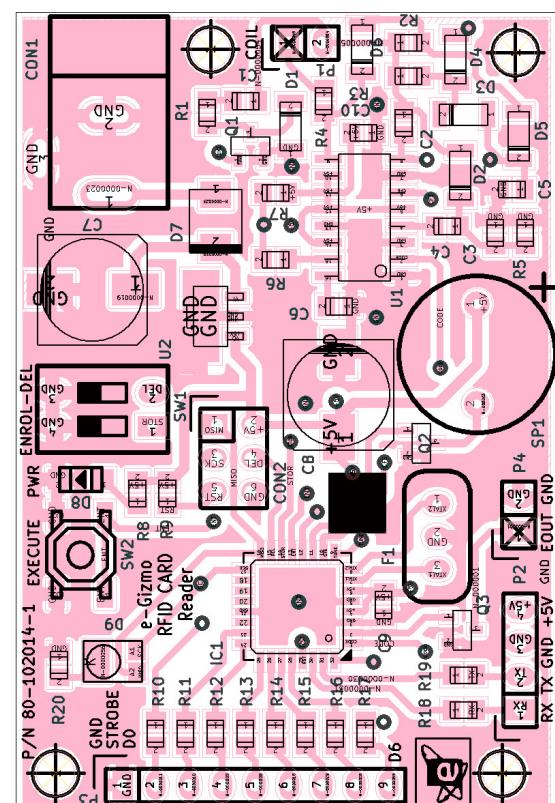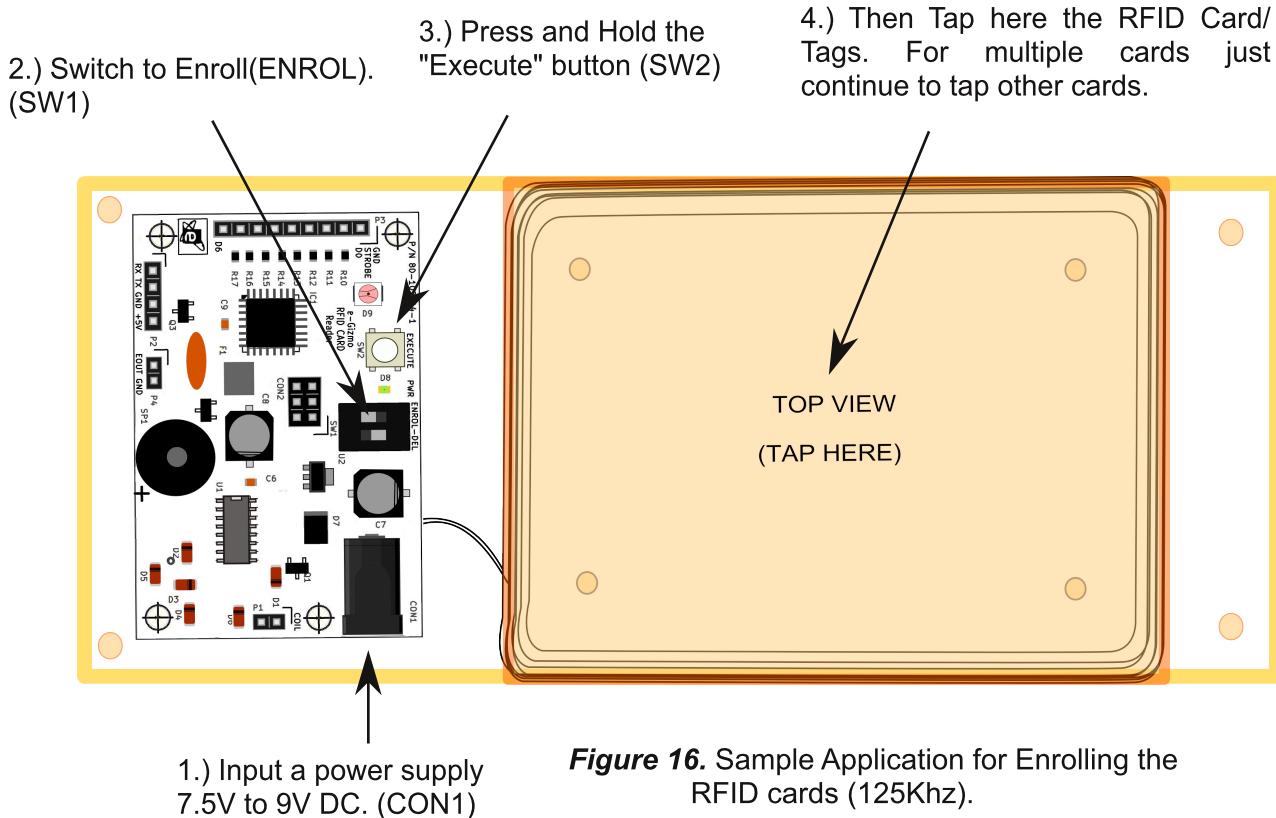
Figure 14. *BottomPCBGuide*
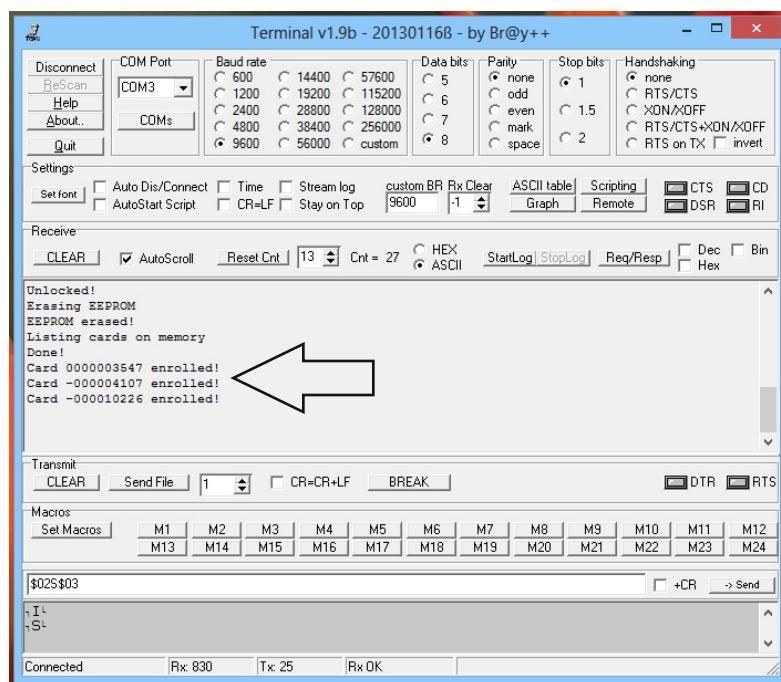
Figure 15. *TopPCBGuide*

How to Enroll a Card or multiple cards:

To monitor the insertion of cards in memory. Open your Terminal in PC using the USB-UART or RS232-TTL converter and set-up the communcation(page 7). (See Figure)

2.) Switch to Enroll(ENROL). (SW1)

3.) Press and Hold the "Execute" button (SW2)

4.) Then Tap here the RFID Card/ Tags. For multiple cards just continue to tap other cards.

TOP VIEW

(TAP HERE)

1.) Input a power supply 7.5V to 9V DC. (CON1)

Figure 16. Sample Application for Enrolling the RFID cards (125Khz).

Figure 17. Enrolled Cards.


```
Unlocked!
Erasing EEPROM
EEPROM erased!
Listing cards on memory
Done!
Card 0000003547 enrolled!
Card -000004107 enrolled!
Card -000010226 enrolled!
```
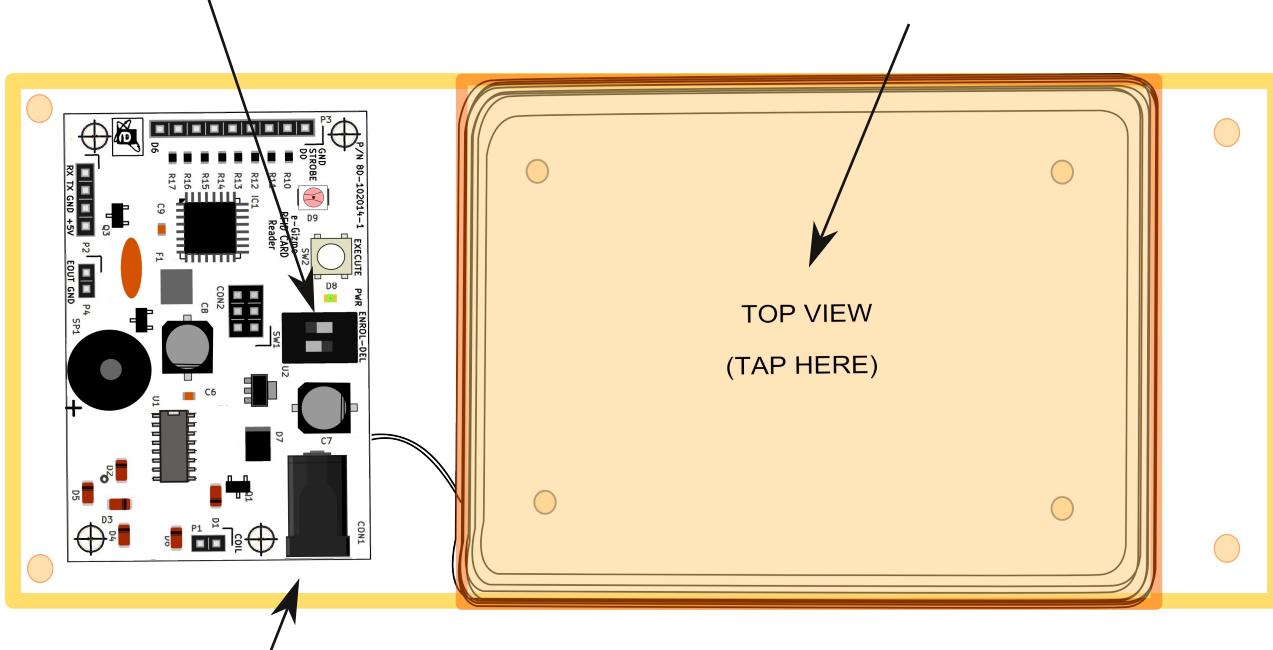
How to Delete a card:
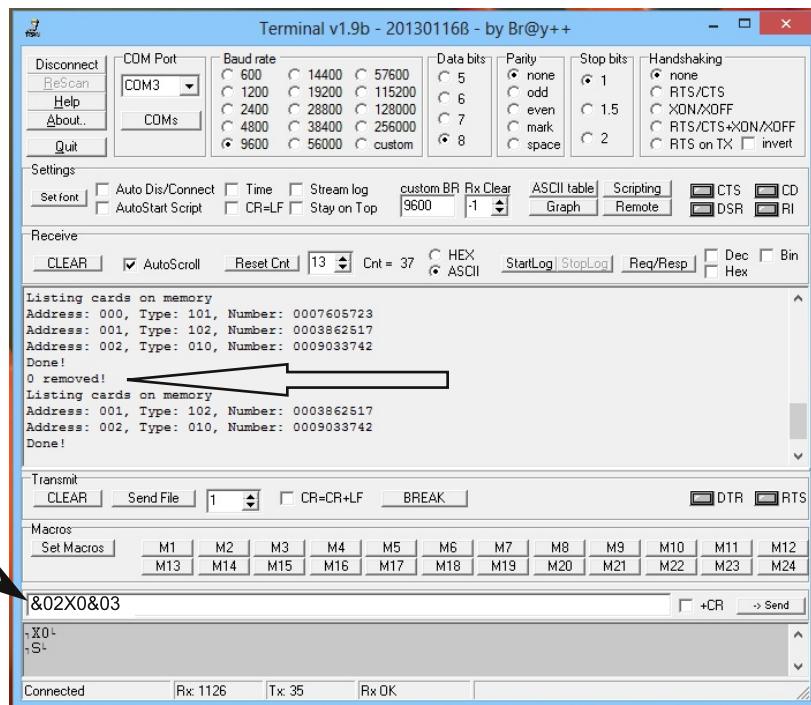
2.) Switch to Delete(DEL). (SW1)

4.) OR Tap here the RFID Card/ Tags to delete.
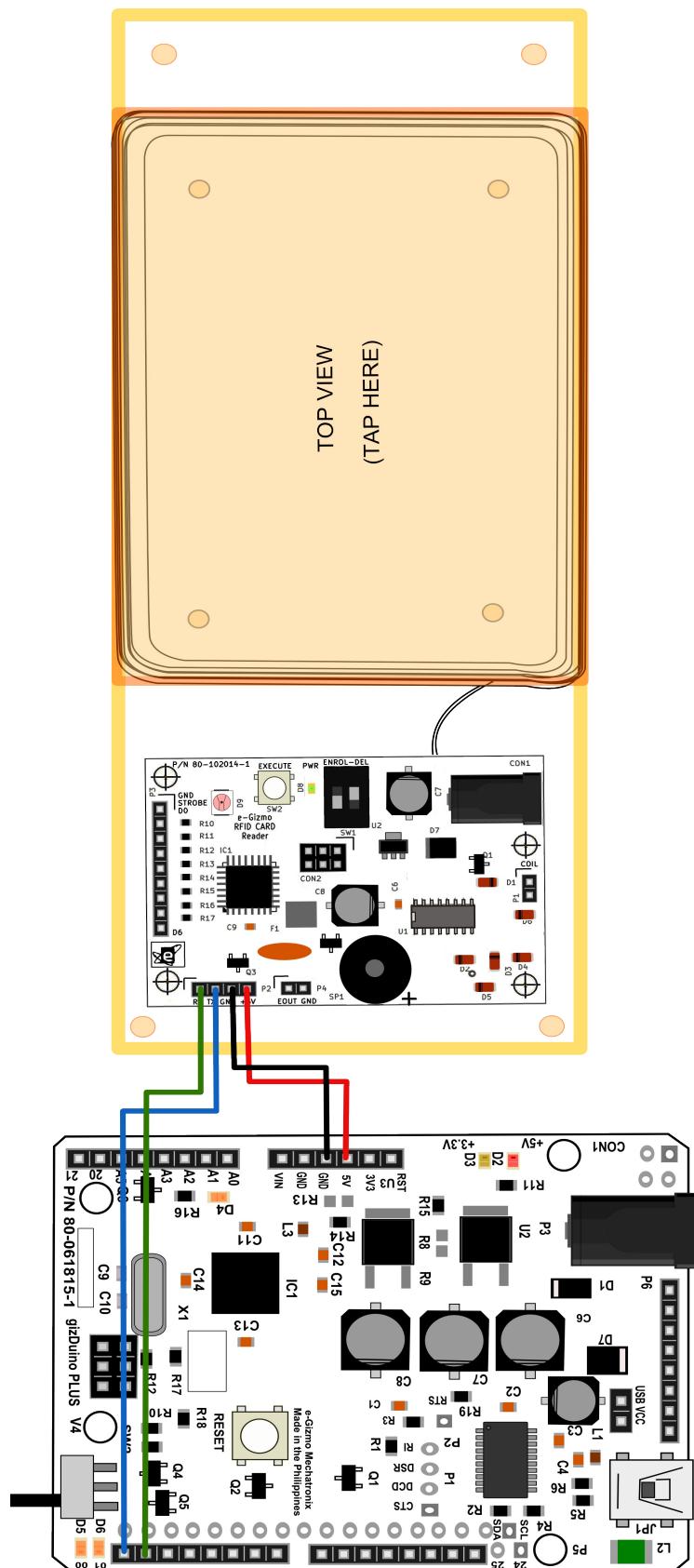

TOP VIEW

(TAP HERE)

1.) Input a power supply 7.5V to 9V DC. (CON1)

Figure 18. Sample Application to delete the cards.

3.) You can type here if you want to delete the card.
Xnn where nn = Card address 0-99.

Figure 19. Deleted Card

TOP VIEW

(TAP HERE)

/* e-Gizmo RFID kit Arduino Demo */

```
// defines used by the serial event
// do not modify
#define  STX 2
#define  ETX 3
#define  SERIALSTX 0
#define  SERIALETX 1
#define  SERIALRDY 2

// RFID read results
 String card_enrolled;   // = 0 if card is not enrolled = 1 if enrolled and valid
 String  card_addr;     // contains valid card index only if card is enrolled.
 String  card_type;     // card type identifier
 String  card_number;   // card serial number

byte  serial_state;
String serialinput;
boolean  carddetected=false;



void setup() {
 // initialize serial:
 Serial.begin(9600);
 pinMode(13,OUTPUT);
 Serial.println("e-Gizmo RFID reader Demo");
 delay(1000);
 Serial.print("\002v0\003");
 delay(500);
 serial_state=SERIALSTX;
 serialinput="";
 check_RFID();

}

void loop() {

  if(check_RFID()==true){

    /* Replace the following demo codes with your own
    */

    // Print the read information
    Serial.print("card enrolled: ");
    Serial.println(card_enrolled);
    Serial.print("card addr: ");
    Serial.println(card_addr);
    Serial.print("card type: ");
    Serial.println(card_type);
    Serial.print("card number: ");
```

```
    Serial.println(card_number);
    Serial.println("");

    // Sample function
    // Notify card is valid if detected as enrolled
    // and momentarily flash pin 13 LED

    if(card_enrolled.equals("1")){
      Serial.print("This card is valid and has index = ");
      Serial.println(card_addr);
      Serial.println("");
      // Flash LED
      digitalWrite(13,HIGH);
      delay(500);
      digitalWrite(13,LOW);
    }
  }

}


/* Do not modify anything in the following codes unless
  you absolutely know what you are doing :-) */

boolean check_RFID(void){

  int  strindex;
  int  strindexe;

    // Read if card data is available
    if(serial_state==SERIALRDY){

    //extract card enrolled portion
    strindex=serialinput.indexOf(",");
    card_enrolled=serialinput.substring(0,strindex);

    //extract card addr portion
    strindex++;
    strindexe=serialinput.indexOf(",",strindex);
    card_addr=serialinput.substring(strindex,strindexe);

    //extract card type portion
    strindex=strindexe+1;
    strindexe=serialinput.indexOf(",",strindex);
    card_type=serialinput.substring(strindex,strindexe);

    //extract card number portion
    strindex=strindexe+1;
    strindexe=serialinput.indexOf(",",strindex);
    card_number=serialinput.substring(strindex,strindexe);
```

```
      // clear received string and ready SERIAL for new stream
      serialinput="";
      serial_state=SERIALSTX;
      return(true);
   }
   return(false);  // return false if RFID data is not available
}


/* This is a interrupt driven serial Rx routine */

void serialEvent() {
  while (Serial.available()) {
    // get the new byte:
    char inChar = (char)Serial.read();

    // Wait for the STX character
    if(serial_state==SERIALSTX){
      if(inChar==STX){
        serial_state=SERIALETX;    // STX character detected, enable next phase
        return;
        }
    }

    // Store rx character to serial input until ETX is detected
    if(serial_state==SERIALETX){
      if(inChar!=ETX){
        serialinput += inChar;
        return;
        }
    }

      // Indicate serial data is ready after ETX is detected
      serial_state=SERIALRDY;
   }
 }
```