Install Guide

TGPI V3.2

BMW K-bike Transmission Gear Indicator

Manual Number: 201509

SAFETY INSTRUCTIONS

Please review the following safety precautions. If this is the first time using this board, then read this manual before installing or using the product. If the product is not functioning properly, please contact Robert MacKinnon for further instructions. Contact details are on the back cover

The lightning symbol in the triangle is used to alert you to the presence of dangerous voltage inside the product that may be sufficient to constitute a risk of electric shock to anyone opening the case. It is also used to indicate improper installation or handling of the product that could damage the electrical system in the product or in other equipment attached to the product.

The exclamation point in the triangle is used to alert you to important operating and maintenance instructions. Failure to follow these instructions could result in injury to you or damage to the product.

Be careful with electricity:

Also follow these precautions:

- **Ventilation**: Blocking the air flow could cause damage. Arrange components so that air can flow freely. Ensure that there is adequate ventilation if the product is placed in the relay box or other enclosed space on the motorcycle
- **Overheating**: Avoid stacking the device on top of a hot component or locating it in the engine bay.
- **Risk of Fire**: Do not place unit on top of any easily combustible material.
- **Proper Connections**: Be sure all cables and equipment are connected to the unit as described in this manual.
- **Water Exposure**: To reduce the risk of board damage, fire or electric shock, do not expose to rain or moisture.
- **Cleaning**: Do not use liquid or aerosol cleaners to clean this unit. Always unplug the power to the device before cleaning.
- **ESD**: Handle this unit with proper ESD care. Failure to do so can result in failure.

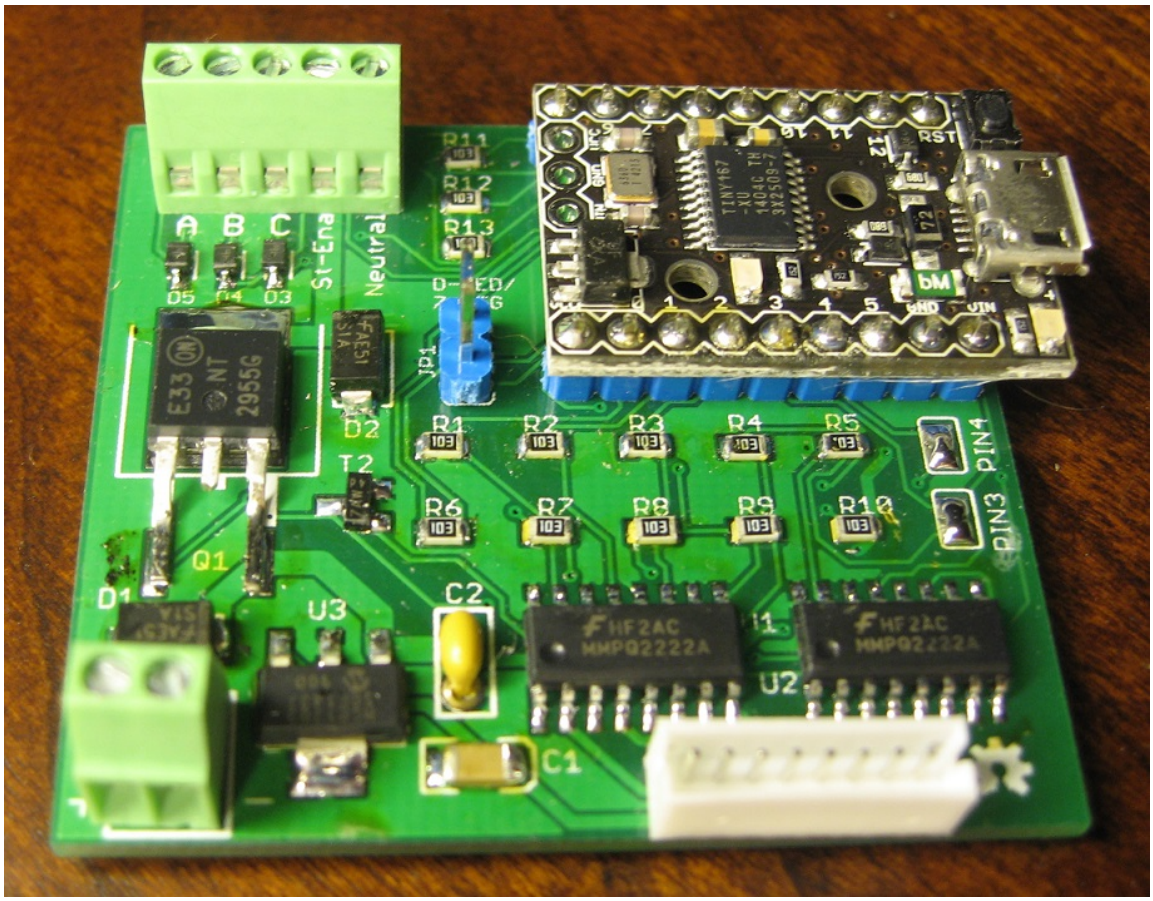
Trademarks

All trademarks in this document are the properties of their respective owners.

TABLE OF CONTENTS

Figures

PACKAGE CONTENTS



Please make sure the following items are included within your package. Contact me if any items are missing or damaged.

- TGPI v3.1 circuit board x 1
- 7-segment Display umbilical cable x 1 OR raw 8-pin JST ZH cable x 1, depending on options selected when ordering
- Documentation Package x 1

INTRODUCTION

The TGPI v3.2 is an interface card used to read the transmission gear position indicator switch on the back of the BMW K-bike / R-bike transmission, and translate that reading into a digit displayed on a variety of display devices. These can consist of:

- 7-segment common anode LED display
- 6 individual LEDs or incandescent bulbs
- Integrated Gear display as part of an aftermarket speedometer/tachometer gauge

The display will indicate gear numbers 1 through 6 and Neutral. In addition to this translation, the circuit detects when the transmission is in the Neutral position and enables the Start Circuit on the motorcycle as well as provides for an off-board Neutral light to be illuminated. At times, positions between gears can be accidently selected (false neutral); the display will be blanked when this condition is detected. This board is ideal a builder / customizer who wishes to replace the BMW instrument cluster with an aftermarket gauge. This circuit works with the following BMW motorcycles:

- K1
- K75
- K100
- K1100
- K1200
- R1100
- R1150
- R1200

Features

- Reads the Transmission Gear Position Indicator switch on the back of the transmission and displays the currently selected gear in an display
- Interface to several types of displays
- Senses neutral gear and enables the start circuit on the bike
- Active-low Neutral light is switched through an on-board transistor
- False neutrals between gears will blank the gear display

Board Connections

Inputs and Outputs

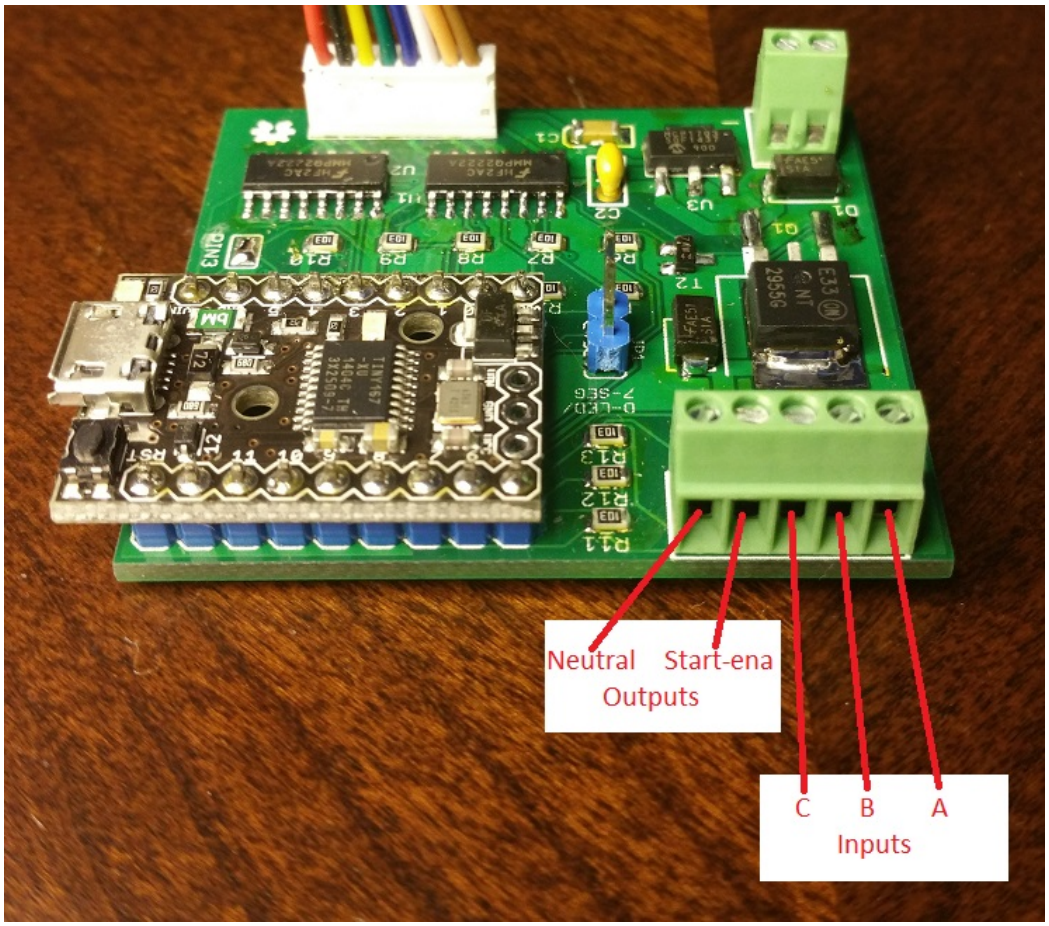


Figure 1 - Inputs and Outputs

- **A:** Input from Yellow/Blue wire on Pin 4 of instrument cluster
- **B:** Input from Yellow/Black wire on Pin 3 of instrument cluster
- **C:** Input from Yellow/White wire on Pin 2 of instrument cluster
- **Start Enable:** Output signal to Black/Green wire on Pin 5 of instrument cluster
- **Neutral:** LED or bulb indicator depending on requirements (see text)

The Neutral output is an active LOW, meaning it will switch to ground when active, otherwise it is open-circuit. If you are using incandescent bulb, generally one side of the bulb will be connected to +12V and the other terminal of the bulb will be connected to this terminal. If you are using LED indicators, the anode of the LED will be connected to +5V and the cathode of the LED will be connected to this terminal. The anode on a LED usually is the longer of two leads on the package.

⚠ Make sure to include a current limiting 220Ω resistor in series with the LED so that it does not get destroyed. The maximum current that this transistor can handle is 500mA.

The Start-enable output goes to the Black/Green wire on the motorcycle. When neutral is selected in the transmission, there will be +12V on this terminal. It is able to source up to 1.2 Amperes of current, which should be twice what is needed by the start relay and other components on that same circuit.

The TGPI switch inputs indicate the gear by grounding each of the three wires in a pattern that creates a binary number between 0 and 7. The Yellow / Blue wire from the TGPI switch is the least significant bit (LSB) and is connected to the "A" terminal. The Yellow / White wire is the most significant bit (MSB) and is connected to the "C" terminal. The Yellow / Black wire from the TGPI switch is the middle bit and is connected to the "B" terminal.

The operation of the switch is according to the following table (an “X” represents a contact made within the switch to ground):

C	B	A	Gear
X	X	X	0 (Neutral)
X	X		1
X		X	2
X			3
	X	X	4
	x		5
		X	6
			Blanks display

Digispark Programming

The on-board computer is continuously running a program that determines the behavior of the circuit. This program could be updated to include new functionality or fix program bugs. To facilitate this change, the on-board

computer can be reprogrammed through a USB port but requires pre-work on the board to prepare it.

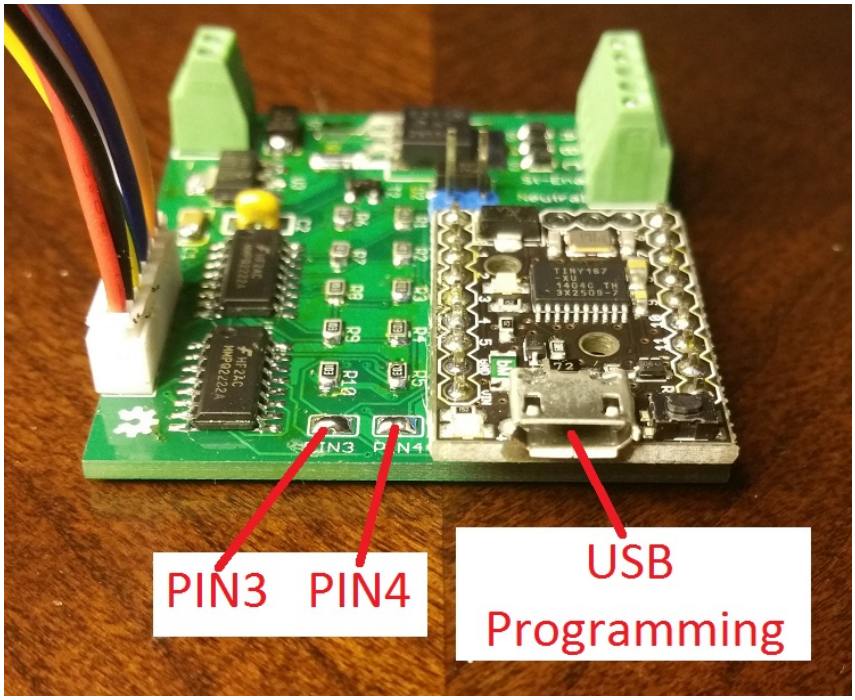


Figure 2 - Programming USB

- **USB for programming:** This plug on the Digispark Pro AVR is used to perform that programming. This takes a mini-USB cable and will hook up to a PC running the Arduino Interactive Development Environment (IDE). The source code of the program running on the AVR is found in Appendix B.
- **PIN3 / PIN4:** Remove to program the Digispark Pro (see text)

The solder bridges on PIN3 and PIN4 are normally not touched. If it becomes necessary to re-program the Digispark Pro AVR, these solder bridges must be removed. Use a fine-tipped soldering iron to melt the solder and copper braid to soak up the molten metal. Place the braid between the pad and the iron, heat the braid and gently rub to remove all traces of the solder. Do not overheat the pads during this operation otherwise the copper traces can be lifted from the board. When cleaned, the pads underneath will no longer be in contact and the computer will be able to be programmed. Once the programming is complete, the pads must again be bridged with solder to enable normal operation. Use a fine-tipped soldering iron and electronic solder to form a small solder bridge between the pads. Do not overheat the pads otherwise the copper traces can be lifted from the board.

Display Connectors and Display Types

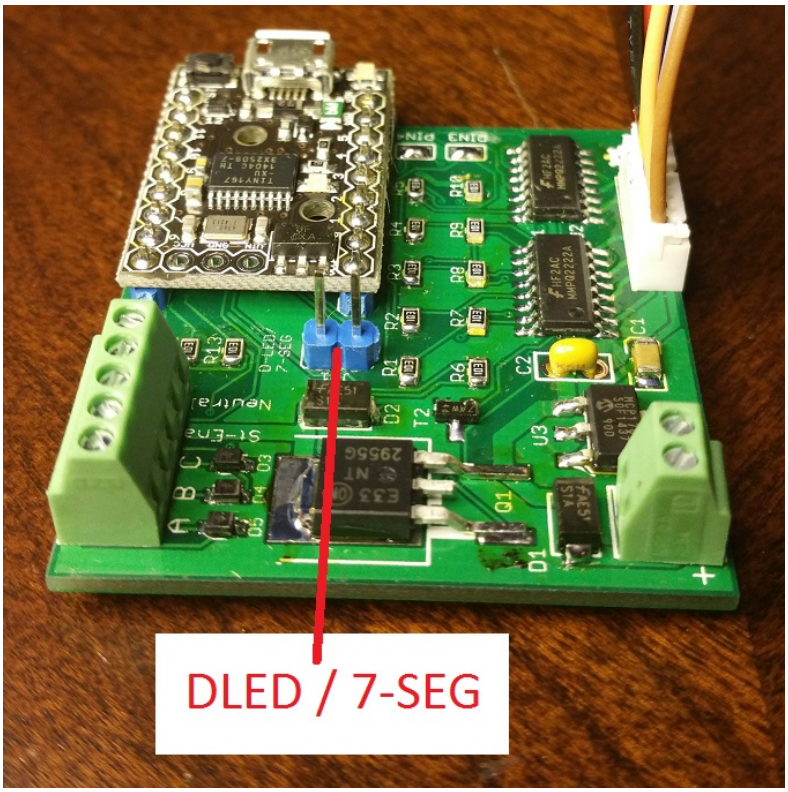


Figure 3 - On-board Jumpers

- **DLED / 7-SEG Jumper:** Selects either 6 discrete LED-type displays or 7-segment LED displays (see text)

⚠ Be sure to set the jumper to match the display to which the board will be connected, to ensure correct operation.

Jumper Installed: With the jumper in place, the routines to drive a 7-segment display are selected. Use the jumper if the 7-segment common anode LED display is being used.

Jumper Removed: With the jumper removed, the routines to drive discrete indicators or a speedometer with integrated gear display are selected. Remove the jumper if 6 discrete indicators or an integrated gear display are being used.

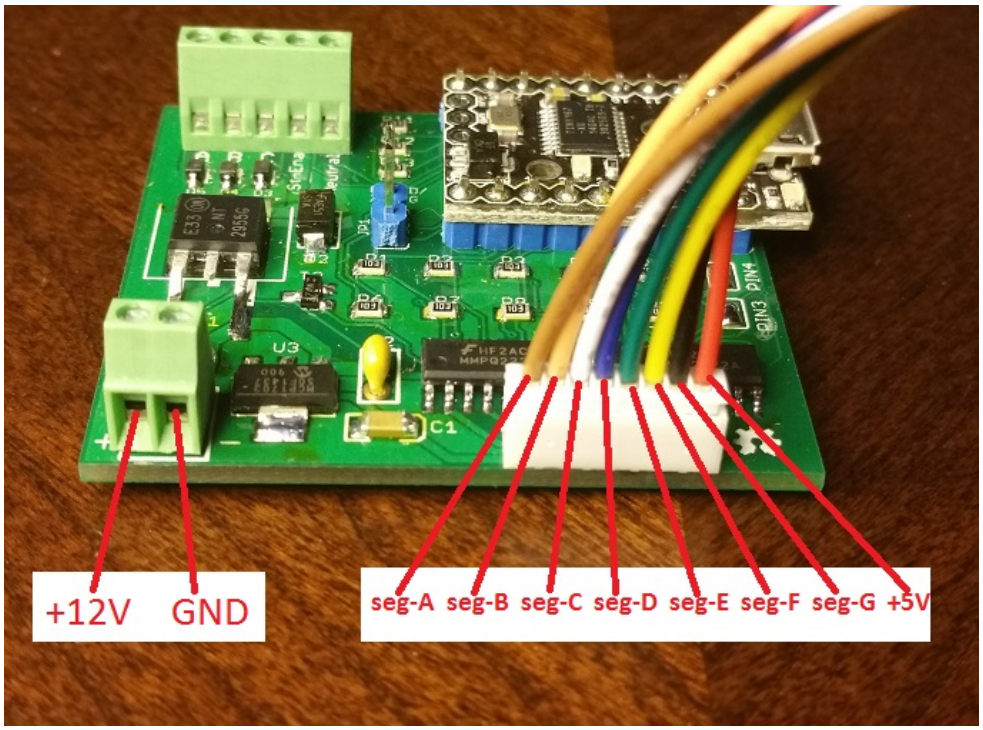


Figure 4 - Power and Display Connectors

- **12V DC:** Connect to the switched +12V DC power of the bike through a 220uH inductor (not supplied)
- **Ground:** Frame ground and TGPI brown wire, Pin 13 of instrument cluster

7-Segment LED Display Type

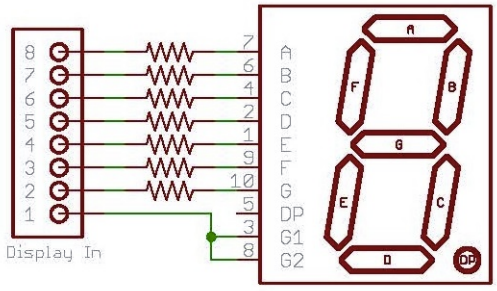


Figure 5 - 7 Segment LED

When connecting a 7-segment display, all 8 pins on the display connector are in use.

- **seg-A (Pin1):** Segment A pin on display
- **Seg-B (Pin2):** Segment B pin on display
- **Seg-C (Pin3):** Segment C pin on display

- **Seg-D (Pin4):** Segment D pin on display
- **Seg-E (Pin5):** Segment E pin on display
- **Seg-F (Pin6):** Segment F pin on display
- **Seg-G (Pin7):** Segment G pin on display
- **+5V (Pin8):** Connect to common pin on display

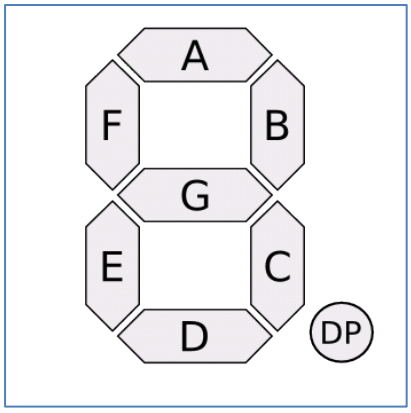


Figure 6 - 7 Segment Display Identification

Discrete Display Type

There are three types of discrete displays that can be connected to the TGPI V3.2.

- 6 LEDs with a common anode
- 6 incandescent bulbs
- An aftermarket gauge with 6 gear input wires (for example, Koso 12000RPM gauge)

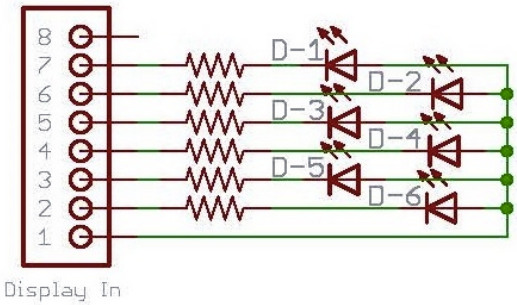


Figure 7 - Discrete LEDs

When connecting the 6 LEDs with common anode, the common anode lead is connected to Pin 1 and each individual LED's cathode is connected to Pins 2 through 7. Pin 8 is not connected. The power from Pin 1 drives the LEDs.

⚠Make sure to include a current limiting 220Ω resistor in series with each LED so that it does not get destroyed. The maximum current that can be handled by the on-board transistors is 500mA.

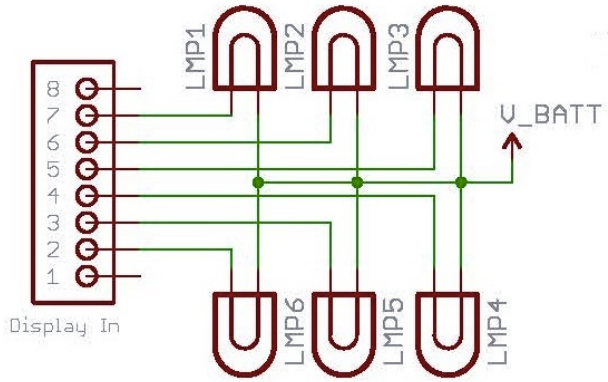


Figure 8 - Discrete Incandescent Bulbs

When connecting the 6 incandescent bulbs, choose low wattage 12V automotive bulbs of a suitable design. The +5V on-board the TGPI v3.2 should not be used to drive the bulbs. Instead connect one side of the filament of each bulb to a common point and supply the bulb with +12V from the bike's electrical system. Connect each of the remaining ends to Pins 2 through 7. Pin 8 is not connected.

⚠ The maximum current that can be handled by the on-board transistors is 500mA. Make sure to select incandescent bulbs which will not draw more than this amount of current.

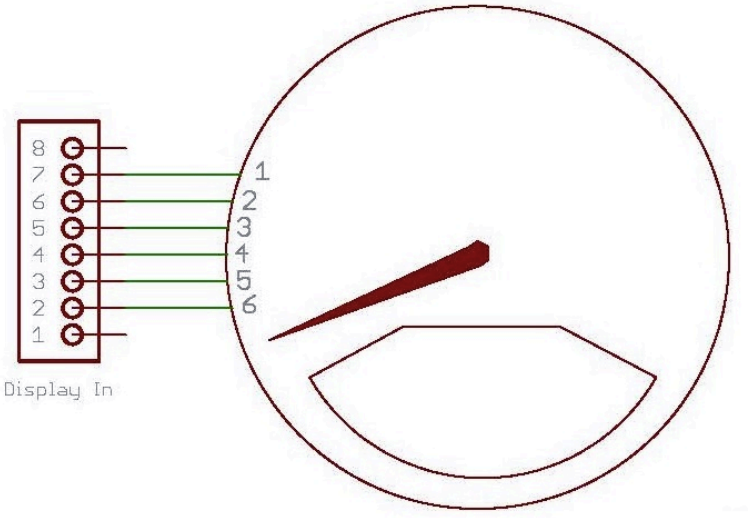


Figure 9 - Aftermarket Gauge with Gear Display

When connecting an aftermarket gauge with built-in gear indicator, these gauges are designed to have one wire for each of the indicated gears. Connect the wires

on the gauge to Pins 2 through 7 on the TGPI v3.2. Pins 1 and Pin 8 are not connected.

Connection Explanation

Step 1: Locate a switched power wire on the motorcycle and a suitable ground wire. Wires up to 18GA can be inserted into the terminal blocks. Connect ground to Ground terminal on the board (marked on the board with a "-" sign). Connect the power to +12V terminal (marked on the board with a "+" sign) through a 220uH inductor. The inductor will help to absorb any transients on the power line and protect the circuit card. Cover the wire connection to the inductor with heat shrink tubing to prevent shorts. The wire of the inductor will be inserted into the terminal block. Clip the wires short so that chances of a short circuit are minimized.

Step 2: Locate the TGPI switch wires on the K-bike's instrument cluster connector (Yellow/Black, Yellow/White and Yellow/Blue). Connect each lead of the TGPI switch to its respective terminal on the board. Be certain to match the TGPI lead colour to the appropriate terminal on the board. This step is important otherwise the indicated gears will be wrong. Also connect the TGPI Brown wire to frame ground; it is normally isolated and not connected to ground at the switch.

Step 3: Locate the Start-enable wire on the harness (Black/Green) and connect that wire to the Start-Enable terminal. Make sure there are no stray strands of wire that might cause a short circuit. A good tip is to tin the end of the wire before inserting it in the terminal block. The Start-enable is active HIGH, meaning it will go to +12V potential when neutral is detected.

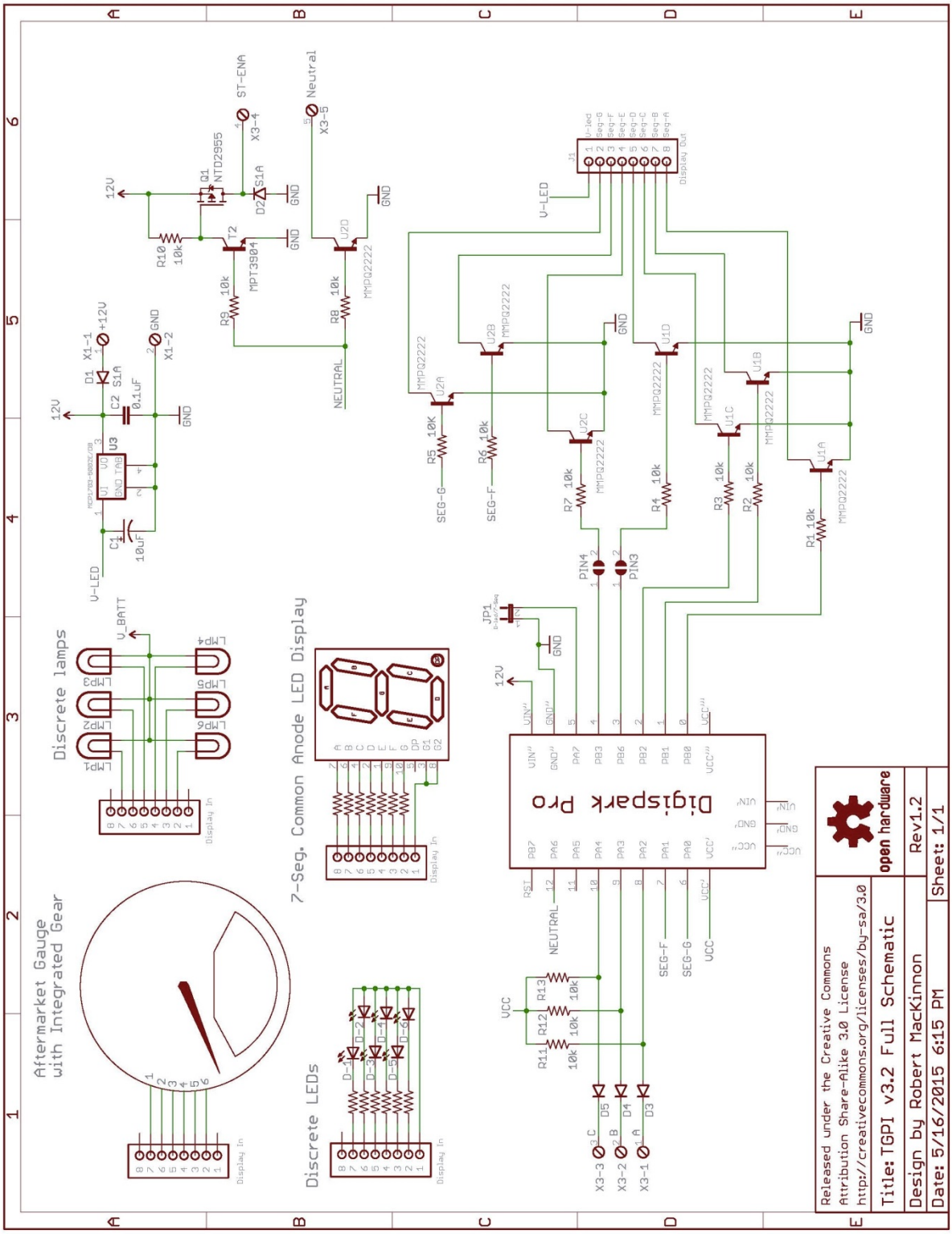
Step 4: Locate the wires for the neutral indicator on your bike. Either they are part of an after-market instrument, or you are creating your own from components. The neutral is active LOW, meaning it will go to ground potential when neutral is detected.

Step 5: Install the display. It should be protected from the elements by locating it in an enclosure, even though it is sealed. If the umbilical was shipped to you with no conformant material encasing the pins, it would be a good idea to use epoxy or silicon

sealant to encase the pins and add rigidity to the wires where they are soldered to the display. Route the supplied umbilical cable through the motorcycle and connect the JST connector to the circuit card. If you are connecting to an aftermarket speedometer with an integrated gear display, refer to the manufacturer's documentation to identify the corresponding terminals on the gauge as well as refer to the schematic diagram below to understand how to connect the corresponding wires from the display connector.

.

Appendix A - Schematic Diagram



Appendix B - Digispark Source Code

```
/*
// [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
// []  tgpi - An enhanced display for the BMW Classic K-bike
// []  Digispark Pro platform:
// []      1. Reads TGPI switch on the Classic K transmission,
// []      2. Simulates start enable and Neutral light
// []      2. Display gear number on external 7-segment LED display or ...
// []      3. ... on discrete LEDs and lamps ...
// []    4. ... Also on aftermarket speedos with integrated gear display

CHANGE HISTORY
2.1 - initial version
3.0 - port to Digispark Pro. RBM 27/Jan/2015
3.1 - addition of OC drivers for higher current. RBM 1/Mar/2015
*/

// [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
// []  C O M P I L E R   D I R E C T I V E S
// [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
// define this to draw in serial terminal debugging messages
// #define DEBUG
// define this if compiling for Arduino UNO test platform
// #define UNO

// [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
// []  L I B R A R Y   I N C L U D E S
// [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]


// [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
// []  S T A T I C   V A R I A B L E S
// [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
#ifdef DEBUG
#define SERIAL_SPEED_BAUD 19200
#endif
```

```
// ------------------------------ Control & Interrupt --------------------------
// target rate = 1Hz  = (1s)/(6.4e-5 s)-1 = 15624
// target rate = 20Hz = 780 and 10Hz = 1561
#define TIMER1_RESOLUTION 1561
// ---------------------------  Gear Switch  --------------------------------
#define GEAR_NEUTRAL 0
#ifdef UNO  // arduino UNO test platform
#define DIGITALPIN_TGPI0          13    // LSB of GPI switch on digital Pin 13
#define DIGITALPIN_TGPI1          12    // GPI switch on digital Pin 12
#define DIGITALPIN_TGPI2          11    // MSB of GPI switch on digital Pin 11
#define DIGITALPIN_NEUTRAL_OUT    10    // Neutral LED on digital Pin 10
#define DIGITALPIN_DIGIT_PIN      9    // display common return
#else   // Digispark Pro platform
// Warning: Don't use Pin 5 for sourcing / sinking current.
#define DIGITALPIN_TGPI0          8     // LSB of GPI switch on digital Pin 12
#define DIGITALPIN_TGPI1          9     // GPI switch on digital Pin 8
#define DIGITALPIN_TGPI2          10    // MSB of GPI switch on digital Pin 7
#define DIGITALPIN_NEUTRAL_OUT    12    // Neutral LED on digital Pin 2
#define DIGITALPIN_DIGIT_PIN      11     // display common return (not used)
#endif
// -------------------------  Display Indicator  ------------------------------
#ifdef UNO  // arduino UNO test platform
#define DIGITALPIN_JP1    0   // Read display type on digital Pin 0
#else   // Digispark Pro platform
#define DIGITALPIN_JP1    5   // Read display type on digital Pin 12
#endif
#define  DISPLAY_7SEG   0   // 7 segment LED display
#define  DISPLAY_DLED   1   // discrete LED display
/* Use defines to link the hardware configurations to the correct numbers
#define  COMMON_CATHODE  0
#define  COMMON_ANODE    1
#define  N_TRANSISTORS   2
#define  P_TRANSISTORS   3
#define  NP_COMMON_CATHODE  1
#define  NP_COMMON_ANODE    0
#define  DASH  11  */
```

```
// [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
// []  G L O B A L   V A R I A B L E S
// [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]

// ------------------------------ GEAR POSITION --------------------------
volatile short currGear = 0;     // 0 = Neutral, or 1-6
volatile short lastGear = -2;    // Force initial update
short displayType = DISPLAY_DLED;  // Default: discrete LED display

// -------------------------  LEDS  -------------------------------------
// 7 segment Arduino pins Seg-A through Seg-G.
#ifdef UNO  // arduino UNO test platform
  volatile static byte segmentPins[] = {2, 3, 4, 5, 6, 7, 8};  // arduino uno
#else   // Digispark Pro platform
  volatile static byte segmentPins[] = {0, 1, 2, 3, 4, 7, 6};  // digispark pro
#endif
  volatile byte digitCodes;
  byte segmentOn;
  byte segmentOff;


// [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
// []  M A I N   S E T U P   A N D   L O O P
// [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
/// ---------------------------------------------------------------------------
/// Setup - gets called once when the Arduino starts
/// ---------------------------------------------------------------------------
void setup()
{
#ifdef DEBUG
  // Setup Serial connection
  Serial.begin( SERIAL_SPEED_BAUD );
#endif
// prepare pins depending on function
  pinMode(DIGITALPIN_NEUTRAL_OUT, INPUT);  // set high impedance mode
  pinMode(DIGITALPIN_TGPI0, INPUT);  // prep for input from tansmission switch
```

```
  pinMode(DIGITALPIN_TGPI1, INPUT);
  pinMode(DIGITALPIN_TGPI2, INPUT);
  pinMode(DIGITALPIN_JP1, INPUT_PULLUP);  // prep to read JP1 state
  findDisplayType();

// turn all segments off
  for (byte segmentNum=0 ; segmentNum < 7 ; segmentNum++)
  {
    pinMode(segmentPins[segmentNum], OUTPUT);
    digitalWrite(segmentPins[segmentNum], segmentOff);
  }

// set the initial gear display
  readGearPosition();
  setNewNum(currGear);

// set Interrupt Service Routine timer
    cli();        // disable global interrupts
    TCCR1A = 0;    // set entire TCCR1A register to 0
    TCCR1B = 0;    // same for TCCR1B

    // set compare match register to desired timer count:
    OCR1A = TIMER1_RESOLUTION;
    // turn on CTC mode:
    TCCR1B |= (1 << WGM12);
    // Set CS10 and CS12 bits for 1024 prescaler:
    TCCR1B |= (1 << CS12) | (1 << CS10);
    // enable timer compare interrupt:
    TIMSK1 |= (1 << OCIE1A);
    // enable global interrupts:
    sei();}

/// -------------------------------------------------------------------------
/// Main loop routine - gets called all the time in an infinite loop
/// -------------------------------------------------------------------------
void loop()
{
```

```
  refreshDisplay(); // Must run repeatedly
}


// [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
// []  I N T E R R U P T   S E R V I C E   H A N D L E R S
// [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]

ISR(TIMER1_COMPA_vect)
{
 readGearPosition();
 setNewNum(currGear);
}


// [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
// []  G E A R   P O S I T I O N   S W I T C H   R O U T I N E S
// [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]

/// -------------------------------------------------------------------------
/// readGearPosition - Determins the current gear from the K-bike gear position switch
/// -------------------------------------------------------------------------
// read TGPI switch. When in neutral, turn on Neutral light and enable start circuit
// Arduino digital inputs are configed to be pulled up externally to +5V.
// TGPI switch outputs inverse BCD code (An "X" indicates a connection to ground (0))
// Yellow/White is MSB and Yellow/Blue is LSB.
//       Yellow/ | Yellow/ | Yellow/
//       White   |  Black  |  Blue
//               |         |
//Neutral    X    |    X    |    X       000
//-----------------------------------
//1st       X    |    X    |           001
//-----------------------------------
//2nd       X    |         |    X       010
//-----------------------------------
//3rd       X    |         |           011
//-----------------------------------
```

```
//4th           |    X    |    X       100
//-----------------------------------
//5th           |    X    |            101
//-----------------------------------
//6th           |         |    X       110
//-----------------------------------
void readGearPosition ()
{
  currGear = 7;  // assume all switches open and then read TGP switch to calclate real gear
  if (!digitalRead(DIGITALPIN_TGPI0)) currGear-=1;
  if (!digitalRead(DIGITALPIN_TGPI1)) currGear-=2;
  if (!digitalRead(DIGITALPIN_TGPI2)) currGear-=4;
  if ( lastGear == currGear)
    return;
//set neutral/start enable accordingly
  if (currGear == GEAR_NEUTRAL) {
    pinMode(DIGITALPIN_NEUTRAL_OUT, OUTPUT);  // set as output
    digitalWrite(DIGITALPIN_NEUTRAL_OUT, HIGH);  // active HIGH turns on MOSFET driver
  } else {
    pinMode(DIGITALPIN_NEUTRAL_OUT, INPUT);  // set high impedance mode
  }
#ifdef DEBUG
  Serial.print( F( "currGear: " ) ); Serial.println( currGear );
#endif
  lastGear =  currGear;
}




// [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
// []  D I S P L A Y    R O U T I N E S
// [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]

/// -------------------------------------------------------------------------
/// findDisplayType - Determine type of display connected -- 7-segment, discrete LED
/// Read on-board jumper. no jumper = DLED; jumper installed = 7SEG
/// -------------------------------------------------------------------------
```

```
void findDisplayType()
{
  displayType = digitalRead(DIGITALPIN_JP1);  // read JP1
  segmentOn = HIGH;
  segmentOff = !segmentOn;

#ifdef DEBUG
  Serial.print( F( "DisplayType reading: " ) ); Serial.println( displayType );
#endif
}



/// ------------------------------------------------------------------------
/// setNewNum - Changes the number that will be displayed.
/// ------------------------------------------------------------------------
void setNewNum(byte numToShow)
{
   //Turn all lights off
   for (byte segmentNum=0 ; segmentNum < 7 ; segmentNum++)
   {
     digitalWrite(segmentPins[segmentNum], segmentOff);
   }

   switch(displayType)
   {
     case DISPLAY_7SEG:
       setDigitCodes(numToShow);
       break;
     case DISPLAY_DLED:
       setLEDCodes(numToShow);
       break;
     default:
       break;
   }
}
```

```
/// ------------------------------------------------------------------------
/// setDigitCodes - Sets the 'digitCodes' that are required to display the input numbers on 7-segment
/// ------------------------------------------------------------------------
void setDigitCodes(byte digits)
{
  // The codes below indicate which segments must be illuminated to display
  // each number.
  static const byte digitCodeMap[] = {
    B00111111, // 0
    B00000110, // 1
    B01011011, // 2
    B01001111, // 3
    B01100110, // 4
    B01101101, // 5
    B01111101, // 6
    B00000000, // 7
    B00000000, // 8
    B00000000, // 9
    B00000000, // BLANK
    B01000000    }; // DASH

    digitCodes = digitCodeMap[digits];
}

/// ------------------------------------------------------------------------
/// setLEDCodes - Sets the 'LEDCodes' that are required to display the input numbers on discrete
LEDs
/// ------------------------------------------------------------------------
void setLEDCodes(byte digits)
{

  // The codes below indicate which segments must be illuminated to display
  // each number.
  static const byte digitCodeMap[] = {
    B00000000, // 0
    B00000010, // 1
    B00000100, // 2
```

```
   B00001000, // 3
   B00010000, // 4
   B00100000, // 5
   B01000000, // 6
   B00000000, // 7
   B00000000, // 8
   B00000000, // 9
   B00000000, // BLANK
   B00000000   }; // DASH

  // Set the LEDCode for each digit in the display
   digitCodes = digitCodeMap[digits];
}



/// -------------------------------------------------------------------------
/// refreshDisplay - Refresh displayed number
/// -------------------------------------------------------------------------
void refreshDisplay()
{
   // Illuminate the required segments for this digit
//   digitalWrite(digitPin, digitOn);
   for (byte segmentNum=0 ; segmentNum <= 8 ; segmentNum++)
   {
     if (digitCodes & (1 << segmentNum))
     { // Check a single bit
       digitalWrite(segmentPins[segmentNum], segmentOn);
     }
   }
}
```

Appendix C - PCB Layout

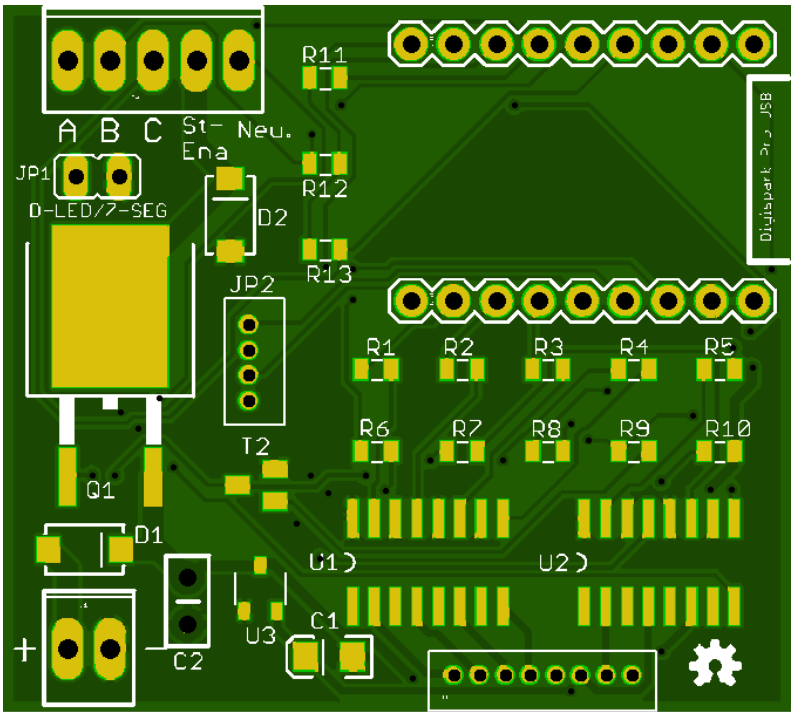


Figure 10 - TGPI Circuit Card - Top

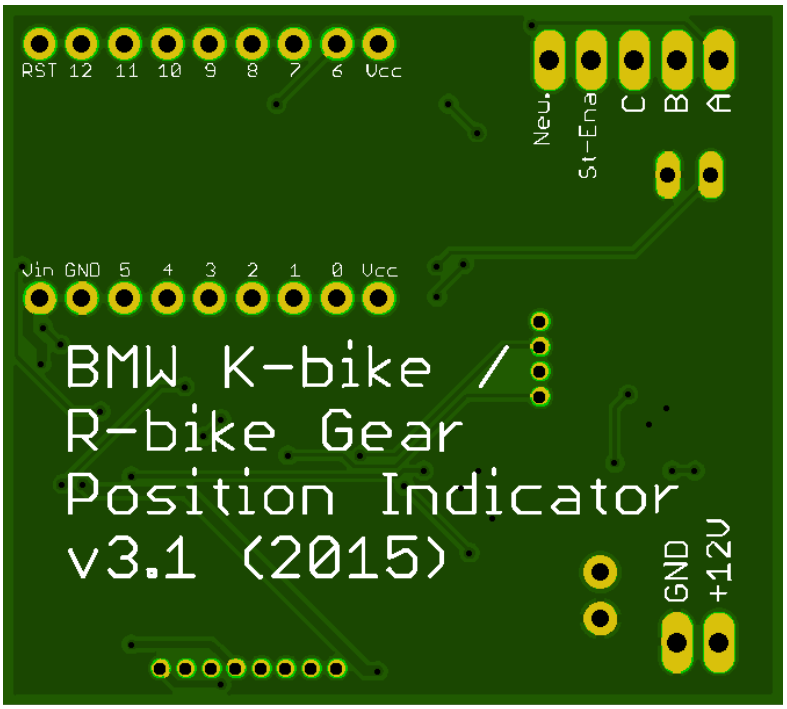


Figure 11 - TGPI Circuit Card - Bottom

TGPI v3.2

Robert MacKinnon

Toronto, Ontario

Email: rbrt.mackinnon@gmail.com

September 2015, document v1.1