Robotized semiautomatic motorcycle transmission development. Electronic and software design

Mihai Neghină[1], *Radu Emanuil* Petruse[2*], *Sebastian* Olteanu[3], *Ioan* Bondrea[2] , *Lucian* Lobonț[2] , and *Gabriel* Stanciu[1]

[1]Lucian Blaga University of Sibiu, Faculty of Engineering, Computer Science Department, Sibiu, Romania
[2]Lucian Blaga University of Sibiu, Faculty of Engineering, Industrial Engineering and Management Department, Sibiu, Romania
[3]Continental Automotive Systems, Sibiu, Romania

Abstract. In this paper, we propose an electrical design (implemented on a PCB board) and an accompanying software design for controlling the automatic gear change. The designs complement the mechanical solutions developed in Part 1. The paper also analyses the issues encountered during the intermediate steps of the development of the electronic module, which is expected to be small and adaptable enough to be installed on a motorcycle without changing its ergonomics. The control software runs on the Arduino Nano board and is built as a state machine with one idle state, five active states that cover different stages of the gear change and one error state for preventing malfunctions in case of an unexpected event. The sketch uses 5,760 bytes (or 18%) of program storage space and 706 bytes (or 34%) of the dynamic memory.

1 Introduction

This paper is based on a research project whose aim is to robotize a manual motorcycle gearbox. This paper is correlated with the paper "Robotized Semiautomatic Motorcycle Transmission Development. Mechanical Design and Integration". To control the mechanisms described in part one of this research, an electronical module must be developed. This module must be small and adaptable enough to be installed on a motorcycle without changing its ergonomics. Also, it must provide robustness in harsh electrical conditions of indoor and outdoor environment and it must assure a linear functionality no matter what loads are applied to it.

This module must be able to:

- Control two direct current motors capable of actuating the clutch and the gearbox lever.
- Separate the high current circuit, which is needed for the two DC motors, from the low current circuit needed for the microcontroller.
- To read feedbacks from the motorcycle's sensors and the robotized mechanism.
- Offer an intuitive human machine interface (HMI)
- Offer a high debugging capability

* Corresponding author: radu.petruse@ulbsibiu.ro

- Be adaptable to different mechanisms

The code compiled for the electronical module must be capable of offering a high reliability no matter what loads (e.g. user commands, false sensor readings etc.) are applied. It must run in a sequential manner and not enter in an infinite program loop or a dead-end (error). In order to show in which, state the program has reached or what problem has encountered, the module can communicate with the computer via USB.

2 Electrical development

2.1 Electrical schematics design

The schematic of the electronic module is made up of blocks, each one with its own purpose. Each block is arranged in a such way that the final populated printed circuit board is easy to debug and offers the best layout compatibility. The functionality of each block should not influence any other blocks, other than those connected to it. All the electronic components are chosen in order to be easy to mount and use. Next, each block will be explained as of its functionality and basics.

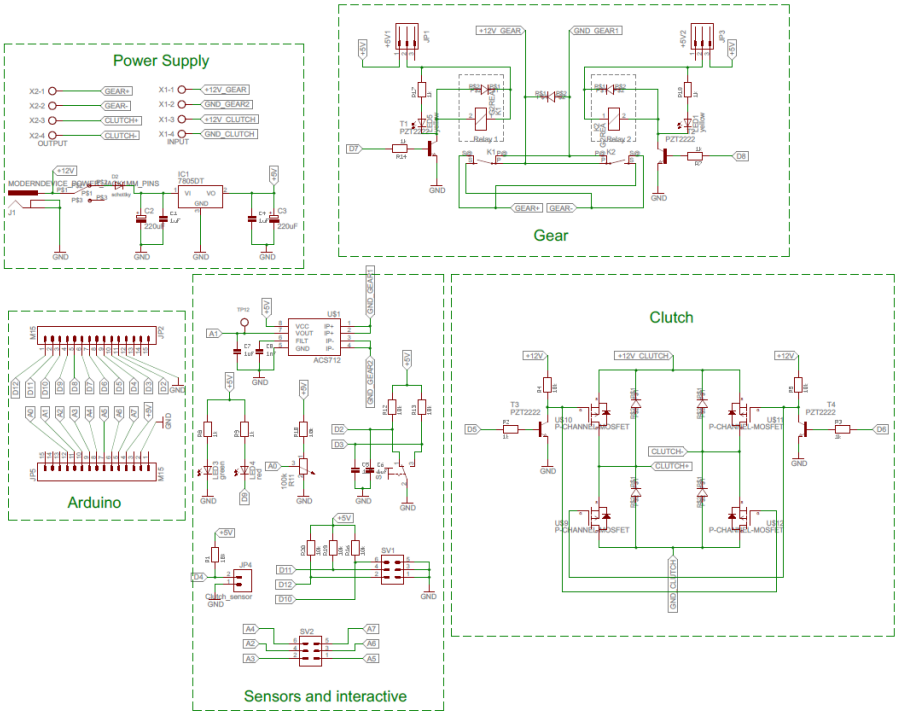

Fig. 1. Electronic schematic

2.1.1 Power supply

There are 3 main power supplies that come directly from the motorcycle battery:
+12V for supplying the logic circuitry from a barrel connector.
+12V_GEAR for supplying the gear shifter dc motor
+12V_CLUTCH for supplying the clutch actuator motor

All supplies are isolated from each other to remove the electric noise from the high current cables of the dc motors. The 3 ground connections are independent from each other.
The printed circuit board (PCB) terminal blocks (two 1x4 blocks) are used to connect cables for the 2 motors, which support a maximum current of 24 amperes. They are also used to grip the supply cables.

The barrel connector is chosen due to its easy assembly on the circuit board and because it can rotate around its interior positive pin.

A gliding switch is used to turn on/off the supply of the logic circuit. It can stand a maximum current of 2 amperes.

An LM7805 (DPAK capsule) voltage regulator is used to convert the ~12 volts from the battery to a constant 5 volts supply for the Arduino and the logic circuits. It offers a supply current of around 1 ampere without a heatsink mounted on top. It is accompanied with 2x220uF electrolytic capacitors and 2x1uF ceramic capacitors on the input and output to remove any unwanted voltage ripple and high frequency signals.

The two direct-current motors (for the gear shifter and the clutch) can reach 3000 rpm, at 12 volts, with a maximum power of 114 watts. Due to its high fly back voltage, to prevent it from suddenly interrupting the power supply, four fly back diodes are used to protect the H-bridge transistors and two fly back diodes are also mounted on the terminals of the two relays.

2.1.2 Arduino Nano

To control the integrated circuit is a ATmega328p microcontroller with a 32 Kbytes of Flash memory, 1 Kbytes EEPROM, 2 Kbytes SRAM that works with a maximum clock frequency of 20 MHz (16 MHz for Arduino Nano). There are two 8-bit Timers and one 16-bit Timer, six PWM channels, 8-channel 10-bit ADC, SPI, USART, I2C interfaces, Watchdog timer, On-chip analogue comparator, Interrupt and Wake-up in Pin Change. The uC can be supplied from 1.8 - 5.5 volts and can work in normal functions in a temperature between -40 -> +105 Celsius grades.

The Arduino's pin connection are set up as follows:

- D2 and D3 pins are connected to the rocker switch S1, which can offer external interrupts. Denounce capacitors of 1uF are used for better reading of signal.
- D4 is used to read the position of the clutch handle. Essentially, it is a Normal-Open micro switch.
- D5 and D6 pins are connected to the transistors that are controlling the clutch, via Timer1 PWM function.
- D7 and D8 pins are used to control the 2 relays. Special hardware control functions are not required in this case, so the program will be used for simple input/output states.
- D9 pin is used for red the LED light error. It can be controlled by PWM.
- D10 pin will be used for an eventual reading of the rotations per minute signal from the motorcycle engine.
- D11 and D12 pins will be used to read the 2 end stop micro switches for the dc motor mounted on the gear. There are pull-up resistors, so when an end stop is pressed, Arduino Nano reads a value of ~0 volts.
- A0 pin is used to read the value of the analogue voltage on the R11 potentiometer.
- A1 pin is used to read the electric current on the gear dc motor. It uses an ACS712 current sensor, which can read a maximum current of 5 amperes

2.1.3 Human-machine interaction

It is done via a rocker button and a potentiometer which calibrates the pulse width modulation for commanding the MOSFET transistors.
There are also 4 LEDs which offer visual information about the system's state:

- 2 orange to display the change in upper or lower gear
- 1 LED POWER green
- 1 red LED in case of error read by Arduino.

Each LED has a current limiting resistor of 1 kOhm. For the safety of the circuit and dc motors, an emergency STOP button is used on the positive supply of the dc motors. It is used in case of an unwanted operation of the motors. Two 4 amp fuses are mounted on the two positive cables of the two dc motors, for a current limitation safety. The 2 amps Schottky diode is used for the essential protection against reversed polarity connection.

2.1.4 Gear control driver

The gear dc motor is supplied via an H-bridge composed of 2 Double-pole Single-terminal Normal-open relays (Relpol RM84-2022-35-1005), which can support a maximum current of 8 amperes on the contacts, at a dc voltage level of 24 volts. The switching speed is around 7 milliseconds, fast enough to give a great speed for the motor command. They are selected due to their wide range of inductor voltage command (3.5 - 12.7 volts) and the possibility of making an H-bridge motor control using just 2 of them.

Due to the supply current of the relay inductor (83 mA) which Arduino's pins cannot provide, an NPN bipolar transistor (PZT2222 in SOT223 capsule) is used for each of the relay to drive their inductor directly from the 5-volt supply. Each transistor offers 600 mA of current.

Two LED indicators (LED1 and LED5) are used to indicate the supply of each relay. Two 1x3 pins (JP1 and JP3) are used to remove any supply voltage of the relay inductors, including the two LEDs.

2.1.5 Clutch control driver

Four P-channel MOSFETs (SUM110P04) are used to create an H-bridge to control the clutch dc motor. The gates of each transistor are connected to +12V signal directly from the dc barrel connector, as to stay open when no command from Arduino is received. The gate-source voltage (Vgs) always must be equal or higher than the drain-source voltage (Vds), so two NPN bipolar transistors (PZT2222) are used for switching from their +12V supply to ground when Arduino wants to close them.

One of the MOSFET transistors can withstand a maximum current of continuous -110 amperes on drain, at a voltage Vds of maximum -40 volts.
Two resistors of 10 kohm are used to limit the current through the two NPN bipolar transistors.

2.2 Layout design

2.2.1 Developing

The module consists of a 35 um thickness copper double-sided board which has all the routes and pads designed using Eagle Cadsoft (trial). The raw photoresistive pcb is developed using a local ultraviolet exposure station in order to obtain best quality of layout, especially for the

0.3 mm traces that are present on the board. Its populated components are mostly of Surface Mount Technology due to the need of pcb space and for better soldering.

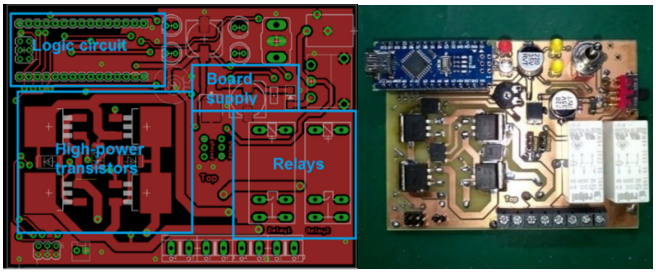

Fig. 2. Developed PCB

2.2.2 Electromagnetic compatibility

In order to limit mutual magnetic influence, traces are separated from each other with a minimum distance of 0.6 mm and the corners of each trace have a slight angle of 45 degrees in order to eliminate the possibility of radio emission. Also, the high-current traces are separated as much as possible from the sensitive low-power traces, and are partitioned in the low side of the board.

3 Control system software development

The code is written in C, using special function in Arduino IDE, specific for Arduino Nano development board. The control of the gear change mechanism is built as a state machine. Sketch uses 5,760 bytes (18%) of program storage space. Maximum is 30,720 bytes. Global variables use 706 bytes (34%) of dynamic memory, leaving 1,342 bytes for local variables. Maximum is 2,048 bytes.

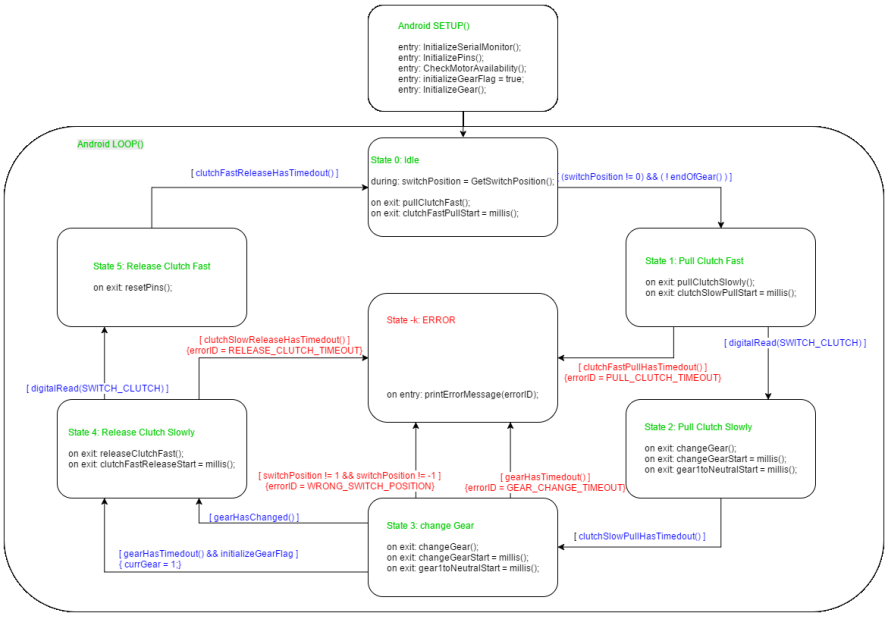

Fig. 3. Finite State diagram

Arduino Setup function

The setup function contains:

- The initialization of the serial monitor (for outputting messages)
- The initialization of the pins
- A check for the availability of the motor
- The setting of gear information. If initialization is requested, the application assumes worst case scenario (MAXGEAR) and attempts to go to neutral. If initialization is not requested, the application assumes that the gear is already in neutral.

Arduino Loop function

The state machine implemented in the Arduino Loop function contains the following states:

- State 0 (Idle) is the state of waiting for a command from the user. State 0 can be exited by a valid user command of changing the gear.
- State 1 (Pull clutch fast) acts on the clutch motor with maximum voltage. State 1 can be exited by reaching a certain position of the clutch (going to state 2) or by timer expiring (going to an error state).
- State 2 (Pull clutch slowly) acts on the clutch motor for a certain duration. State 2 is exited when the timer expires and goes to state 3.
- State 3 (Change gear) acts on the gear motor. State 3 is exited when the timer expires or when one of the end stops indicates that the gear has changed.
- State 4 (Release clutch slowly) acts on the clutch motor with moderate voltage. State 4 is exited when the timer expires (going to an error state) or by reaching a certain position of the clutch (going to state 5).
- State 5 (Release clutch fast) acts on the clutch motor with maximum voltage for a certain duration. State 4 is exited when the timer expires (going to state 0).
- Error states output an error message and prevent any further action.

State 0: Idle

State 0 (idle) is the state of waiting for a command from the user. While in State 0, the application checks the command switch pins, in order to determine whether there is a valid command. If no valid switch command is received, the application considers the user switch to be in the neutral position and remains in State 0.

In case a valid switch command is received and there is a possibility to change the gear in the direction requested by the user, then the cycle of changing the gear begins (the cycle goes through all positive states 1→2→3→4→5→0). Checking the possibility of changing the gear in the direction requested by the user prevents the cycle to begin in case that the motor already is in the maximum gear and the user requests a higher gear, or if the current gear is 0 (neutral) and the user requests a lower gear.

At the initialization phase, the application assumes that the current gear is in MAXGEAR (worst case scenario) and attempts to find the neutral. It does so by requesting lower gears automatically until gear 1 has been identified, and then requests once more a lower gear. During the initialization phase, the application runs several times through the cycle 0→1→2→3→4→5→0 without user intervention and any user command is ignored until the initialization phase has ended.

On exiting State 0, the application starts the timer for State 1, resets all pins and writes the pin for the fast phase of the clutch pull.

State 1: Pull clutch fast

The first phase of pulling the clutch acts on the clutch motor with maximum voltage. The application can exit State 1 in two cases:

- If the timer expires, in which case the application enters error state
- If a pin indicates that the phase has ended (application enters State 2)

On exiting State 1 towards State 2, the application starts the timer for State 2, resets all pins and writes the pin for the slow phase of the clutch pull.

State 2: Pull clutch slowly

The second phase of pulling the clutch acts on the clutch motor for a certain duration. The application exits State 2 when its timer expires. There is no error related to State 2 because there is currently no measurement to indicate whether the clutch has been fully pulled.

On exiting State 2, the application starts the timers for State 3 and resets all pins.

State 3: Change Gear

Depending on the positions of the gears relative to each other and the user command, the function sets the one or the other pin for controlling the gear motor. The implementation described in this paper is specific to the gears being positioned so that neutral is between the first and second gear. The initialization also takes into account this particularity of motorcycles.

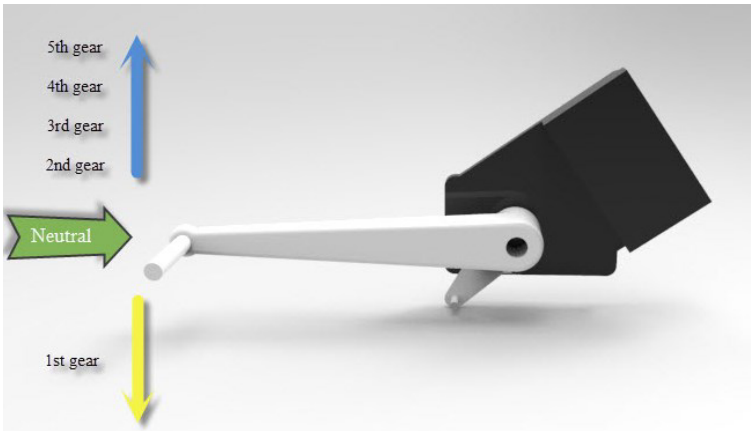

Fig. 4. Motorcycle gears

The application can exit State 3 in three cases:

- The gear has indeed changed (application enters State 4). The indication that the gear has changed relies on:
  - either endstop pin is triggered (the endstop indications are not taken into consideration for the first few milliseconds to avoid unplausible triggerings), or
  - If the user requested a change from gear 1 to neutral, 50ms have passed since entering State 3
- The timer expires in the initialization phase. The expiration of the timer is taken as an indication that the current gear is 1.
- The timer expires outside the initialization phase, in which case the application enters an error state

On exiting State 3 towards State 4, the application starts the timer for State 4, resets all pins and writes the pin for the slow phase of the clutch release.

State 4: Release clutch slowly

The first phase of releasing the clutch acts on the clutch motor with moderate voltage (moderate PWM duty cycle). The application can exit State 4 in two cases:

- If the timer expires, in which case the application enters error state
- If a pin indicates that the phase has ended (application enters State 5)

On exiting State 4 towards State 5, the application starts the timer for State 5, resets all pins and writes the pin for the slow phase of the clutch pull.

State 5: Release clutch fast
The second phase of releasing the clutch acts on the clutch motor for a certain duration. The application exits State 5 when its timer expires. There is no error related to State 5 because there is currently no measurement to indicate whether the clutch has been fully released.
On exiting State 5, the application resets all pins.
State: Error
When entering an Error state, the application outputs a message indicating the error and then moves to the ultimate Error State, where nothing happens (in order to not flood the serial monitor by sending one error message again and again).

4 Control software and electronic intermediate tests

During these tests, some issues with the PCB were discovered. The command for the 2 electrical operations was based on an Arduino development board which commands two SPDT-NO relays. These relays will form two H bridges for the DC motor which will change the gears and four MOSFET transistors for a future adaptation of an electric actuator dedicated for operating the clutch.
During the project development, some circuit design flaws were also encountered.

4.1 Layout issues:

4.1.1. PCB Version 1

After completing the electronic scheme, we reached the PCB development stage, drilling for pins and viases, then in the stage of soldering the components. The images bellow reflects the incomplete PCB in terms of populating the components on the board.
Due to unsuitable design of the PCB, the footprints of some components were wrongly made resulting in the difficulty of soldering their connection pins on the pads and viases of the PCB.

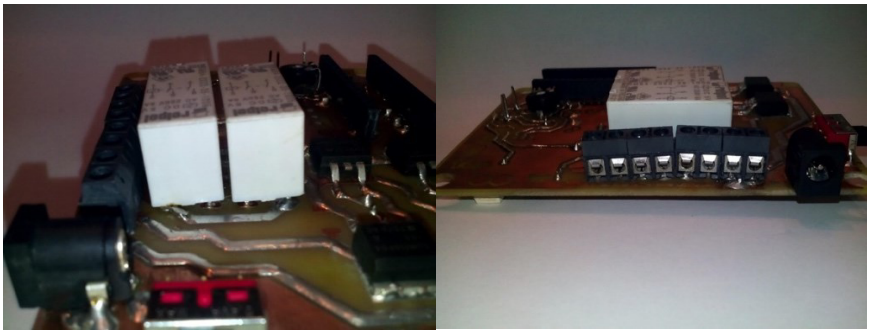

Fig. 5. Poorly welded relays and supply terminals for the board

One of the main reasons for the incorrect placement of the components is guiding current routes of the THT components on the top side of the board, therefore access to pads proved to be difficult. The THT components viases do not have connections between bottom and top side of the board.

Another problem of the circuit encountered during electrical tests is the poorly designed footprint of the voltage regulator.

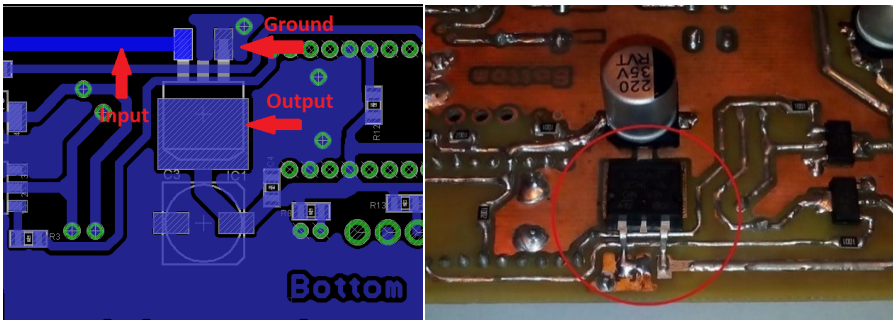

Fig. 6. Voltage regulator LM7805 - layout (footprint is incorrect)

4.1.2 PCB Version 2

P-channel MOSFET transistors were installed with the source to GND_CLUTCH and the drain to +12_CLUTCH, which led to an internal short-circuit on the internal diode.

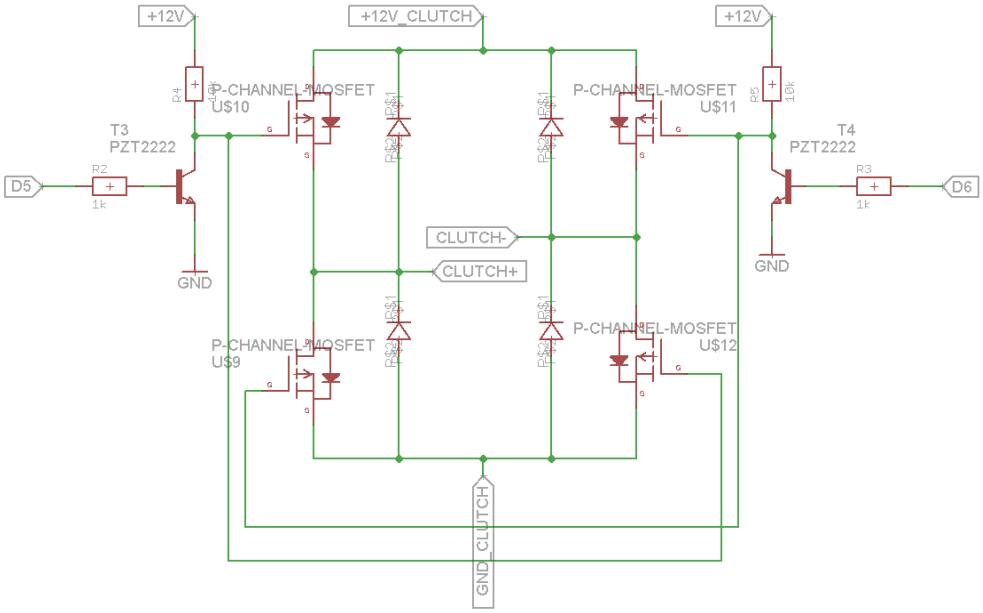

Fig. 7. Schematic with incorrect mounted transistors

Protection diodes D2, D3, D4, D5, D6, D7 and D8 did not have a corresponding footprint pin in the wiring diagram, so that every time you fed the relays and transistors, the short was made at the ground. The problem was later resolved by being mounted in the right direction.

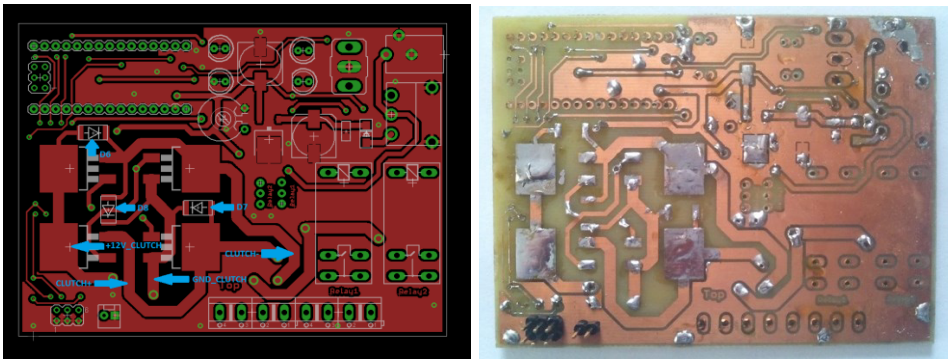

Fig. 8. Incorrect Layout Top

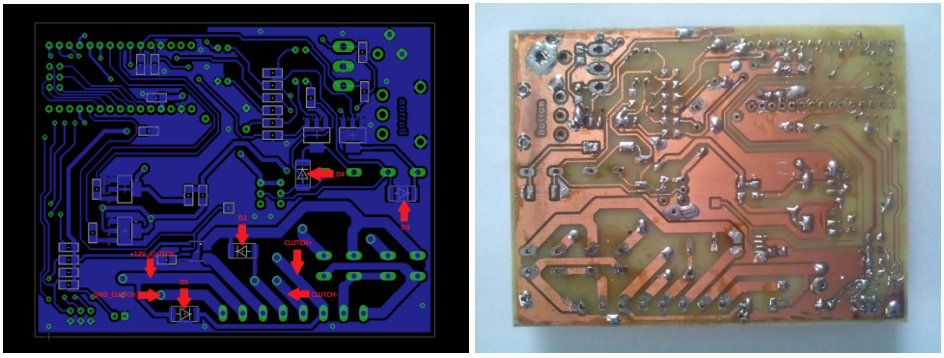

Fig. 9. Incorrect Layout Bottom

Due to not using correct fuse ratings (4a fuse, and we misleadingly used a 25a fuse) for the motors to work and removing power from the electronics, a short-circuit problem appeared during tests. Because the H-bridge for the clutch motor uses P channel MOSFET transistors, a voltage always must be existent on the gate of each transistor to keep it open. Once power was removed from the electronics, the voltage on the gate of the transistor fell to ~0.5, which made the transistor close its source-drain connection and short-circuit the whole 4 transistor H-bridge. This lead to a high current of almost 20 amperes passing through it, which heated up the transistors and the traces. One trace (GND_CLUTCH) heated in an extreme manner and it interrupted itself.

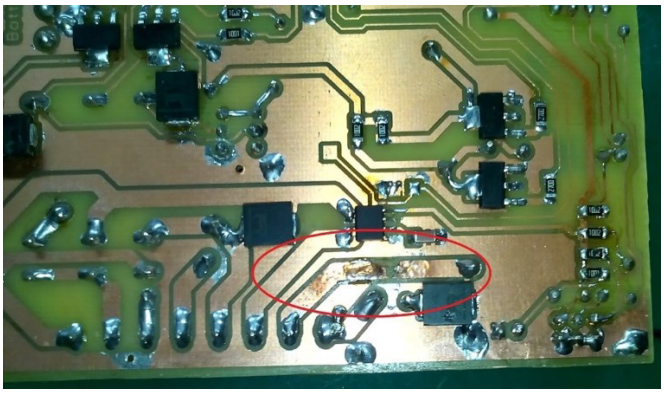

Fig. 10. Interrupted circuit due to short-circuit

5 Electrical functioning

5.1 PCB Version 1

From this point of view, we discovered that supplying for a short time the voltage regulator, voltage on the OUT pin fluctuated between 0.3 and 0.6 volts. Thus, we found that the footprint's pins from the program used to design the layout were not correct. Because of this reason, we discontinued to supply the rest of the circuit.

5.2 Version 2

The power supply was limited to a maximum current of 700 mA for safety. Reverse current diodes have allowed shorting +12V to ground due to the incorrect mounting, but without affecting their functionality, supporting up to 100 A.

After the correct install of the protective 1.5SMCJ120A diodes, we commanded the transistor's gate to rotate the motors, at which moment we observed their inappropriate behaviour due to incorrect mounting. The maximum current given went through the transistor's diodes and the voltage dropped from 12V to ~2V.

5.3 Version 3

Because the high electrical current once passed through the transistors, it heated them up at almost 100 degrees' Celsius temperature, but it didn't affect their functionality, as they are rated for a current of up to 100 amperes. They still performed well after resolving the trace interruption problem with solder.

6 Conclusions

The goal for the research is to eliminate the user's effort of changing the gears and make the gear-change as smooth, fast and autonomous as possible. In the current state in which the motorcycle is mounted on a stand, the project cannot be tested outdoors making this impossible to offer a clear idea of how the system should work in the purposed environment. But, despite all the encountered PCB layout problems, the module behaves as expected and can even be improved for better performance in the near future.

Acknowledgement: The research leading to these results has been supported by a research contract from Continental Automotive Systems Sibiu, Contract no. C-1028 form 17.10.2016.

References

1. M. Simon, *Programming Arduino*: *Getting Started With Sketches* (McGraw-Hill Education edition, 2011)
2. M. Banzi, M. Shiloh, *Getting Started with Arduino*, (Maker Media, Inc; 3 edition, 2014)
3. https://www.arduino.cc/en/Main/ArduinoBoardNano accessed on 17.01.2017