**RoboTrike**

**Functional Specification**

**Description**: RoboTrike allows an operator to control a three-wheeled robotic car via a keypad and display. The system itself consists of two distinct components: (1) a keypad and display through which the operator interacts with the system and (2) the three-wheeled motor unit that can move around under the user’s control. The keypad is composed of 16 keys, each programmed with different operations that help the user to move the RoboTrike manually. Beyond basic operations such as moving right, left, forward and reverse, the keypad includes features such as speed manipulation, set handshake, and turret control. This allows the RoboTrike to move freely in desired directions and speeds while also taking advantage of the turret and laser. Commands are entered from the keypad and transmitted via the serial interface to the RoboTrike motor unit; the motor unit then sends back status information and any other desired data via the same serial interface. The display is responsible for displaying any appropriate messages while the system is running. Such messages include the command being executed and the position of the turret. It also indicates any errors that occur on the serial interface or errors that are reported by the motor unit itself. The user will need to fix the problem accordingly after checking the error messages.

**Global Variables**: None.

**Input**: The RoboTrike is moved manually by an operator using the keypad, which sends commands via the serial interface to the motor unit. Hence, all input is through keys on the keypad:

|  |  |  |
| --- | --- | --- |
| **Key Name** | **Key #** | **Description** |
| Left Angle Up | 1 | Increases the angle with respect to the -x axis (max angle +85o, each click will increase the angle by 5o) |
| Forward | 2 | Moves the RoboTrike forward at the designated speed |
| Right Angle Up | 3 | Increase the angle with respect to the +x axis (max angle +85o, each click will increase the angle by 5o) |
| Turret Up | 4 | Points the turret up until it is +60o from the horizontal (each click will raise the turret by 5o) |
| Left | 5 | Moves the RoboTrike left at the designated angle with respect to the -x axis (angle changed through Key 1/9) |
| Fire | 6 | Fires the turret laser |
| Right | 7 | Moves the RoboTrike right at the designated angle with respect to the +x axis (angle changed through Key 3/11) |
| Position Turret | 8 | Rotates the turret in the counterclockwise direction (each click will rotate the turret 15o) |
| Left Angle Down | 9 | Decreases the angle with respect to the -x axis (minimum angle -85o, each click will decrease the angle by 5o) |
| Back | 10 | Moves the RoboTrike backwards at the designated speed |
| Right Angle Down | 11 | Decreases the angle with respect to the +x axis (minimum angle -85o, each click will decrease the angle by 5o) |
| Turret Down | 12 | Points the turret down until it is -60o from the horizontal (each click will lower the turret by 5o) |
| Speed Up | 13 | Increase the speed of the RoboTrike (maximum speed 100mm/s, each click will increase the speed by 5mm/s) |
| Speed Down | 14 | Decrease the speed of the RoboTrike (minimum speed 5mm/s, each click will decrease the speed by 5mm/s) |
| Set Handshake | 15 | Turns RTS (Ready To Send) / CTS (Clear To Send) handshake on or off (will initially be turned off) |
| Debug Mode | 16 | Displays incoming serial data on the display in hex |

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

Figure 1. The physical arrangement of the 16 keys on the keypad. Each key has a different command associated with it, allowing the user to control the RoboTrike.

**Outputs**:

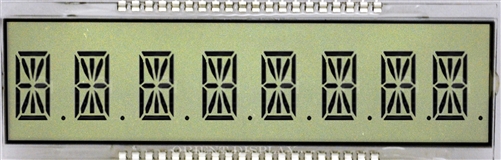


Figure 2. Sample LED display with its 8 digits, each with 14 segments and a right hand decimal point per digit. There will be four modes on the display that show different information as needed.

There are two main outputs for the RoboTrike system, the first being the display and the second being the motor unit. The LED display can produce up to 8 digits, each consisting of 14 segments that can be turned on or off to produce letters and numerals. Compared to the traditional seven-segment display, the RoboTrike system’s 14 segment display is extended by four diagonal and two vertical segments with the middle horizontal segment broken in half. This allows the LED to accurately display certain ISO basic Latin alphabet characters such as “Q” and “V” that otherwise could not have been utilized. With such capability, the LED continually shows necessary messages while the RoboTrike system is operating. There are four alternating display modes: (1) Mode A displays RoboTrike’s direction and speed, along with handshake settings (2) Mode B shows the angle from the positive and negative x axis at which the RoboTrike moves along, (3) Mode C displays information regarding the turret’s position and elevation, and (4) Mode D enters debug mode where incoming serial data is displayed in hex. Mode A is the default display mode and will update according to the user’s input on the keyboard. Mode B will be activated whenever the user changes the angle using Keys 1, 3, 9, and 11. Mode C is turned on whenever the user interacts with the turret, and Mode D is entered when the user clicks on Key 16.

The first three digits of Mode A tells the user RoboTrike’s direction of movement with either “FWD”, “REV”, “LFT” or “RGT” (standing for Forwards, Backwards, Left or Right, respectively); if the RoboTrike is stationary, the previous direction of movement will be displayed. The next three digits display RoboTrike’s current speed (in mm/s) from 000 to 100. The seventh digit is either an “O”or “X” based on the handshake settings; if the RTS/CTS handshake is on, the digit will display an “O”. The last digit will show an “F” if the user is firing the laser on the turret. Hence, if the RoboTrike is moving forward at a speed of 35 mm/s with the handshake and laser on, display Mode A will show “F W D 0 3 5 O F”.

The first digit of Mode B always shows the letter “L”, and the next three digits display the angle at which RoboTrike will move along with respect to the *negative* x axis when moving “l”eft. The second digit will display either “+’ or –“, and the third and fourth digits will display an angle between 00 and 85. The fifth digit will always show the letter “R”, and the last three digits display the angle at which RoboTrike will move along with respect to the *positive* x axis when moving “r”ight. The sixth digit will display either “+” or “-“, and the seventh and eighth digits will display an angle between 00 and 85. Thus, if the RoboTrike is set to move left at an angle +35 from the -x axis and move right at an angle -23 from the +x axis, display Mode B will show “L + 3 5 R – 2 3”.

The first digit of Mode C always shows the letter “T” to let the user know that the system has entered “t”urret orientation mode. The second digit will either be a “+” or “-“ depending on the turret’s position with respect to the horizontal. The third and fourth digits will then display the turret’s angle from the horizontal from 00 to 60. The fifth digit will either be a “+” or “-“ depending on the turret’s position from the front of the RoboTrike. The last three digits will then display the turret’s angle from the front from 000 to 180. Thus, if the RoboTrike’s turret is positioned 32o above the horizontal and is pointing backwards, display Mode C will show “T + 3 2 – 1 8 0”.

Mode D will enter debug mode where incoming serial data is displayed on the display in hex. If there are any, it will also show errors that occur on the serial interface or are reported by the motor unit. The user will have to manually click on Key 16 to enter debug mode, but error messages will pop up automatically. The error messages will start off with the letters “ERROR” followed by the number associated with the error. More details are listed on the Error Handling section.

The motor unit is consisted of the motors themselves and the laser on the turret. Three DC motors are used to rotate the wheels in either direction and move the RoboTrike via PWM. There is one stepper motor that is used to rotate the turret, and the system will utilize only the counterclockwise direction of the motor. One servomotor is used to set the angle of elevation of the laser, which can either be 60o above or beneath the horizontal. All of the motors are controlled via 11 bits of parallel output of an 8255. The laser is controlled via a single bit of parallel output of an 8255. The motor unit and the keypad/display all communicate through a standard serial port using a 16C450 UART.

**User Interface:** The user moves the RoboTrike manually via the preprogrammed keypad that was described in the Input section above. Whenever a command is chosen from the 16-key keypad, it is sent via the serial interface to the motor unit. The motor unit, along with the turret and laser, then responds accordingly and send back status information to the keypad via the same serial interface. The display then lights up to the proper mode (A, B, C or D) and displays current information about the RoboTrike movement. If an error is found on the serial interface or reported by the motor unit, it will automatically show up on the display, allowing the user to act accordingly.

Clicking on Keys 2, 10, 5, 7 (the “movement” keys) will move the RoboTrike forwards, backwards, left and right, respectively. The DC motor drivers are connected to port B of an 8255; it latches the data written to it and holds it on the output lines. Each motor may run clockwise or counterclockwise as determined by one bit of Port B for each motor. To move the RoboTrike continuously, the user must press down on a key; a single click will move the RoboTrike for only half a second. If the user releases any of the movement keys, RoboTrike will respond quickly enough that it will stop within a second. The movement keys can only be pressed one at a time; if the user presses down on Key 2 and accidently clicks on Key 5, RoboTrike will ignore Key 5 and just continue to move forward. If two movement keys are clicked (*not* pressed down) at the same time, RoboTrike will follow the command of the key that communicated with the motor unit first and then follow up with the second command. Keys 13 and 14 (the “speed” keys) can be used simultaneously with the movement keys and will either speed up/down the RoboTrike. Clicking on the speed keys will increase/decrease speed by 5mm/s, and pressing them down will be equivalent to one click for second. That is, pressing down for 5 seconds will be equivalent to 5 continuous clicks. The DC motor speed is set using PWM (Pulse Width Modulation). The movement keys can be used with any other keys except the “angle” keys, as described below.

Since the RoboTrike has delicate cords that cannot be tangled, its direction cannot be changed while moving; that is, the Keys 1, 3, 9 and 11 (the “angle” keys) will only work when the RoboTrike has fully stopped. To change the direction at which RoboTrike moves towards, the user can utilize the Keys 1/9 (which will change the angle at which RoboTrike moves left) and Keys 3/11 (which will change the angle at which the RoboTrike moves right). If the user clicks on an angle key while pressing down on a movement key, the RoboTrike will stop and the display will change to Mode D with an error message. Clicking on Key 16 will enter debug mode that will display incoming serial data in hex.

Clicking on Keys 4, 6, 8, 12 (the “turret” keys) will either control the turret or laser. Key 4 will point the turret up until it is +60o from the horizontal; a single click will move it up by 5o and pressing down will be equivalent to one click per second. Key 12 will point the turret down until it is -60o from the horizontal; a single click will move it down by 5o and pressing down will be equivalent to one click per second. The servomotor is connected to port C of an 8255; it is controlled by a single bit of port C and latches the data written to it and holds it on the output lines. Clicking Key 8 will rotate the turret 15o in a counterclockwise direction and pressing it down will rotate it 15o per second. The stepper motor is also connected to port C of an 8255; it is configured as a unipolar drive and as such has 4 bits controlling it. The motor has a maximum step rate of 50 half-steps/sec. Clicking Key 6 will fire the laser and pressing the key down will fire it once per second; the laser itself is connected to a driver on Port B of an 8255.

The serial interface used in the RoboTrike has a simultaneous transmission and reception of data at the same baud rate. The RTS and CTS lines can be used for handshaking when connected directly to a computer; clicking on Key 15 will either turn this feature on or off. The motor unit is controlled via the serial interface and outputs it current status to the serial interface whenever the status changes. The keypad and display sends commands and receives status updates over its serial interface, while the motor unit receives commands and sends status updates.

**Error Handling:** If there are any errors that that occur on the serial interface or reported by the motor unit, the LED display will enter Mode D and display an error number that specifies what type of error occurred. The debug mode itself displays incoming serial data on the LED in hex, which allows the user to keep track of which data is going in and out of the serial interface. Since there is no feedback in the RoboTrike system (check the Limitations section for more details), errors regarding the distance or direction of the RoboTrike cannot be handled correctly. If there are any other errors, however, the error messages will pop up automatically with the appropriate error number.

Error number 1xx’s have to do with errors that occurred on the serial interface. Since the serial data may be arriving much faster than it can be acted upon by the RoboTrike or the LED display, it is necessary to buffer the serial port data. If the buffer fails, an error message will popup specifying which type of error happened. Errors may also occur when the serial interface is unable to send the status over to the keypad or if the keypad cannot send commands to the motor unit. Some of the serial interface errors include farming (100), parity (101), break (102), overrun (103), or buffer overflow (104).

Error number 2xx’s have to do with errors that are reported by the motor unit. These errors predominately occur when there is a bad command sent to the motor unit from the keypad. Such errors include pressing the movement and angle keys together (200), trying to position the turret beyond the allowed elevation (201), attempting to move the RoboTrike faster than the maximum speed (202), and so on. Some of the 2xx’s are temporary error messages like the 201 and 202; these will disappear after 3 seconds. Others like the 200 will not disappear until the user fixes the problem manually.

**Algorithms:** The two’s complement to decimal conversion algorithm is used to convert a 16-bit signed value to decimal in the Dec2String function. For this algorithm, you first check if the number is negative or positive by looking at the sign bit. If positive, simply covert it to decimal by multiplying 1 or 0 with the corresponding 2n value. If it is negative, make it positive by inverting the bits and adding one. Then, covert the result to decimal as if it were positive, and negate the number to get the desired value.

**Data Structures:** Queues, an abstract data structure, are used for some RoboTrike routines. They have the advantage of being open at both ends, one to insert data and the other to remove data.

**Limitations**: The RoboTrike system has fixed wheels that cannot rotate in any other direction. This hinders the mobility of the RoboTrike and makes it extremely difficult for the user to move the three wheeled robot in a non-linear path. Moreover, a robotic car would normally include some sort of feedback to help determine its position. RoboTrike does not have this feedback system and must take a “dead reckoning” approach to movement. That is, there is no way of telling if the system moved the correct distance or direction.

**Known Bugs**: None.

**Special Notes**: None.