BILL V'S DERBY TIMERS

Arduino Based Pinewood Derby 2-Lane Racetrack Timer

With DerbyNet Race Manager Interface

USER MANUAL

1 INTRODUCTION

The Pinewood Derby racetrack timer discussed herein is a practical “minimal hardware” design that provides the features needed to conduct a successful race event. This version is specifically designed to operate with the DerbyNet™ Race Management software by Jeff Piazza (http://jeffpiazza.github.io/derbynet/) running on a PC to receive, record and display the race results.

The Arduino UNO™ board used in this project provides the interfaces to the optical lane sensors, race start (gate) switch and executes the software that performs the timing, monitors the track status and sends the race results and track status to the PC for post-race processing and display by the DerbyNet™ software. Data is transferred between the Arduino and the PC via the USB interface.

The freely available open source DerbyNet™ race management software is based on a standard web server model that lets you run multiple clients (browsers) in multiple platforms to access the same database. It not only provides for display of the race results but also integrates many features that allow you to manage all aspects of the race event including registering racers and cars, generating schedules, determining awards, providing reports, and more. It is highly customizable making your race more organized and more enjoyable for all.

> ***Note:*** *The free* DerbyNet™ Race Management *software is not provided as part of this package and must be downloaded by the end user at:* http://jeffpiazza.github.io/derbynet/*. Refer to the DerbyNet web site and DerbyNet documentation* for *details and operating instructions.*

2 SOFTWARE INSTALLATION

NOTE: The following instructions are for the installation of the Arduino DerbyNet Timer software consisting of source code file PWD_DerbyNet_Timer02_Ver1.ino.

2.1 Arduino UNO Software Installation

2.1.1 Arduino Software Environment Installation

Download and install the Arduino Integrated Development Environment (IDE). For complete step-by-step instructions on how to download and install the Arduino Integrated Development Environment (IDE), visit https://www.arduino.cc/ . Versions are available for Windows, Mac OS X, and Linux. The environment makes it easy to write code and upload it to the board. The WEB site also provides tutorials to help you every step of the way.

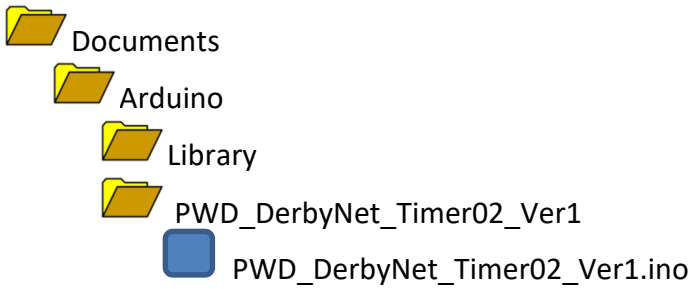
***NOTE**: Installation of the Arduino IDE also installs the necessary USB drivers for your PC to communicate with the Arduino via a serial RS-232 type (i.e. COM1, COM2, etc.) communication link. These drivers are also used by the DerbyNet software discussed in Section 2.2 below.*

2.1.2 Arduino Source Code Installation

Once you have the Arduino development environment installed on your PC (per Section 2.1.1) perform the following to install the Arduino source code:

1. Create a subfolder called 'PWD_DerbyNet_Timer02_Ver1' under the "Arduino" folder. The "Arduino" folder was created during the install, usually under your Documents folder.
2. Extract the file 'PWD_DerbyNet_Timer02_Ver1.ino' from the zip file provided and copy it into the 'PWD_DerbyNet_Timer02_Ver1' folder created in the previous step.

This is a typical folder/file hierarchy but may vary:



2.1.3 Uploading the DerbyNet Timer code to your Arduino Board

Perform the following steps to upload the timer code to the timer unit's Arduino microcontroller. Note that once the software has successfully been uploaded to the Arduino board, it is there permanently, unless overwritten by another upload. Hence, you only have to perform these steps once. Future use of the Arduino timer during your race events does not require an upload. Just connect the Arduino timer to your PC, launch the DerbyNet Race Management software and you should be ready to go.

Step	Action	Comments
1	Ensure your PC is powered up and ready.	
2	Connect the track timer (Arduino board) to your PC USB port via the USB cable.	Arduino board power indicator illuminates.
3	Double click on the 'PWD_DerbyNet_Timer02_Ver1.ino' file to launch the file and bring up the Arduino integrated development environment.	Arduino integrated development environment window is displayed with the PWD_DerbyNet_Timer02_Ver1 source code listing.
4	Select the 'Tools/Board' pull-down menu to select/verify the Arduino/Genuino Uno board is selected.	
5	Select the 'Tools/Port' pull-down menu to select/verify the COM port selection (i.e. COM1, COM2, etc.).	Arduino COM port is selected / verified.
6	Select "→" (upload) to start the compile and upload process.	The program will compile and automatically upload to the Arduino board.
7	**OPTIONAL** To verify a successful upload, perform the diagnostic test procedure located at the end of this document.	

2.1.4 The Auto Reset Feature

This version of the Arduino timer code contains an auto-reset feature that when enabled, automatically readies the timer for the next race upon completion of the previous race. The software is delivered with this feature enabled. To disable this feature perform steps 1 thru 3 of the above procedure to bring up the timer source code in the Arduino integrated development environment. Locate the following line of code in the 'Setup' section of the source code and change the value from 'true' to 'false' as shown below. Use the 'save' command to permanently save your changes.

```
FROM: Auto_Reset = true;        //Enables timer to automatically reset…
TO:   Auto_Reset = false;       //Enables timer to automatically reset…
```

Note that when this feature is disabled the user must press the reset button to ready the timer for the next race.

3 DERBYNET RACE MANAGEMENT SETUP

***Note**: It is recommended that the user install and become at least somewhat familiar with the DerbyNet software before proceeding, especially the Timer Interface module (derby-timer.jar).*

Once DerbyNet is launched, perform the Setup procedures per the DerbyNet documentation. For this timer, ensure the following options are set accordingly (Refer to Fig.1 below):

- Number of lanes: 2
- Track Length (in feet): tbd (start line to finish line)
- Displayed time precision: 5 digits

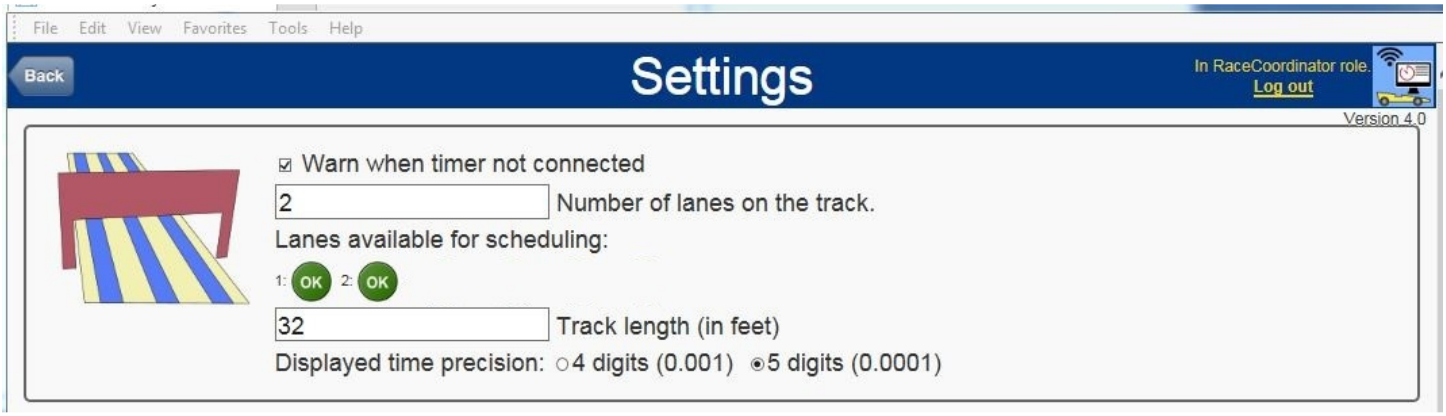


Figure 1. DerbyNet Setup

3.1.1 Connecting the Timer to DerbyNet

The DerbyNet race management software is capable of automatically detecting a number of commercially available timer systems (refer to the DerbyNet documentation) including the Arduino based “MiscJunk PDT timer" (www.miscjunk.org/mj/pg_pdt.html).

DerbyNet communicates with the track timer through a small java program called ‘derby-timer.jar’ (part of the DerbyNet software package). Refer to your DerbyNet documentation for accomplishing this. Note that you may need to install Java onto the computer to which the timer is connected. When ‘derby-timer.jar’ is launched it will try to auto-detect and establish communication with your timer. It should connect as the “MiscJunk PDT” timer (See Fig. 2).

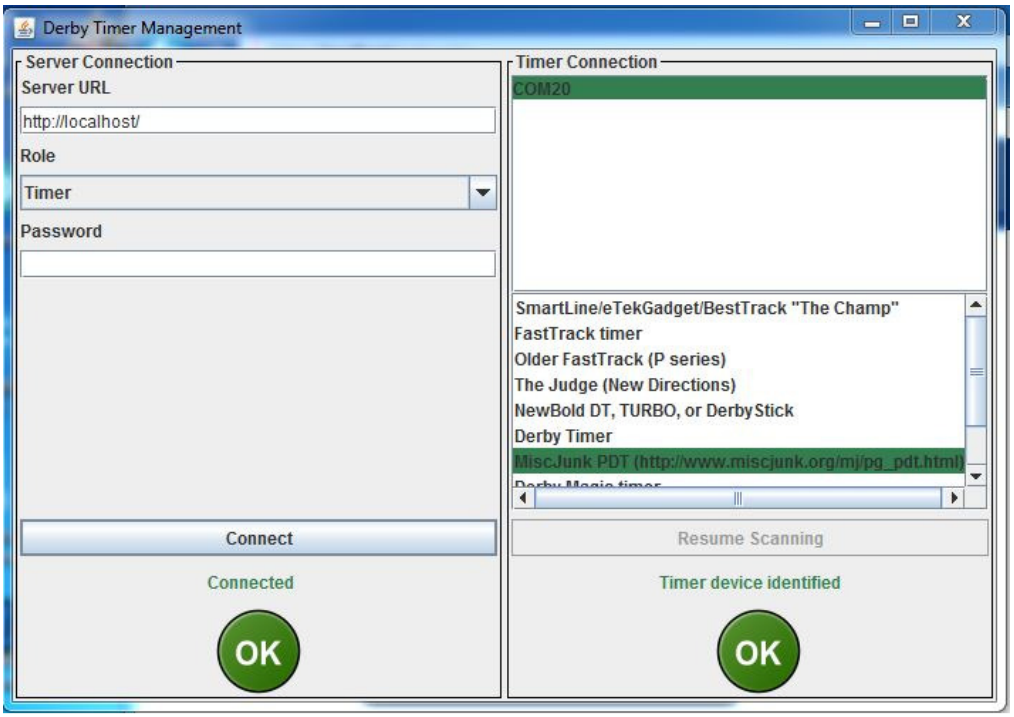


Figure 2. DerbyNet Setup

4 TIMER OPERATION DESCRIBED

4.1 Arduino Timer Operation Overview

As mentioned earlier, this version of the Arduino timer software has been specifically coded to work with the DerbyNet Race Management software. This was accomplished by modifying the Arduino timer code to mimic the “MiscJunk PDT timer" communication data format. Single letter commands are used to pass track status and control messages between the timer and the DerbyNet software. Table 1 below summarizes these commands.

Table 1. Arduino Timer Serial Data Control Codes

Char.	Source	Description
G	From DerbyNet	Check Gate request – Sent by DerbyNet to determine if the timer is ready (See ‘O’ command)
R	From DerbyNet	Reset timer request – Sent by DerbyNet to issue a timer reset
M	From DerbyNet	Lane mask request – Sent by DerbyNet to mask a given lane. This command is followed by the lane number (i.e. ‘M2’ will request the timer mask lane 2)
U	From DerbyNet	Lane Unmask request – Sent by DerbyNet to command the timer to unmask all lanes.
O	From Timer	Gate Open – Response to Check Gate request if the Gate switch is open (Not Ready). If gate is closed the timer’s response will be the “.” (Acknowledge character).
.	From Timer	Acknowledge – Response to various DerbyNet requests.
K	From Timer	Timer Ready – Response to a Reset request if the timer is in the ‘Ready’ state
B	From Timer	Timer Start – Sent when timer transitions from Ready to Racing (Gate switch released).
V	From DerbyNet	Version request – Sent by DerbyNet to request timer software version. The Arduino timer responds by sending the message “vert=1.00”
N	From DerbyNet	Number of Lanes request – Sent by DerbyNet to request timer number of lanes. The Arduino timer responds by sending the message “numl=x”, where x is the number of lanes
D	To Timer	Special (Non DerbyNet) “debug” command that can be issued from a serial monitor program. Will cause the Arduino timer to respond with a message giving the state of the gate switch and optical lane sensors. See the diagnostic test procedure at the end of this document.

Operation of the Arduino based timer is straight forward. The track timer code running on the Arduino cycles through four states as follows:

- Track Not Ready
- Ready
- Racing
- Finished

The “Track Not Ready” state is entered at initial power-up. The “Track Not Ready” state is also entered when the Arduino timer receives a reset command and it senses that one or more of the optical lane sensors is obstructed or the Gate Switch is in the wrong state to start the race. In this case it will send the character “O”

to the PC running DerbyNet to indicate it is not ready to race. The red LED will be illuminated when the timer is in the “Not Ready” state.

The “Ready” state is entered when the timer receives a reset command and the optical lane sensors and Gate Switch are in the correct state to start the race. When transitioning to the “Ready” state the timer will send the character “K” to the PC running DerbyNet to indicate it is ready to start. The yellow LED will be illuminated when the timer is in the “Ready” state.

The “Racing” state is entered when the timer has sensed activation of the Gate Switch indicating the cars have left the gate and the race is underway. When transitioning to the “Racing” state the timer will send the character “B” to the PC running DerbyNet to indicate the race is underway. The green LED will be illuminated when the timer is in the “Racing” state.

The “Finished” state is entered when all cars have crossed the finish line or when 10 seconds have elapsed, whichever occurs first. The 10-second time-out is for cases where a car fails to cross the finish line or a lane was not used. The 10-second time-out does not apply for lanes that have been designated as BYE lanes (i.e. masked) and thus ignored by the timer. The timer will send the lane times to the PC running DerbyNet in the following format:

1 – A.AAAA(cr/lf)
2 – B.BBBB(cr/lf)
Where: A.AAAA and B.BBBB are the times.
Note that for lanes that timed-out (car did not cross) or are masked, a time of 9.9999 seconds will be sent. This ensures that cars that did not finish get correctly ranked in the standings.

After the timer has sent the race results to the PC and if the auto-reset feature is enabled, it checks the status of the gate switch and the optical lane sensors. And when they are in the correct state to start the next race, the timer will automatically advance from the “Finished” state to the “Ready” state. If the auto-reset feature is not enabled, the user must press the reset button to advance the timer out of the “Finished” state.

DIAGNOSTIC TEST PROCEDURE

This test procedure was written to assist in testing and troubleshooting the Arduino based timer hardware and software.

Setup:

- Bring up the Arduino Integrated Development Environment (IDE) on your PC (Refer to Section 2.1)
- (Optional) Select and load the PWD_DerbyNet_Timer02_Ver1.ino file.
- Select the 'Tools/Port' pull-down menu to select/verify the port selection (i.e. COM1, COM2, etc.).
- Open the serial monitor by clicking on the little magnifying glass near the upper right corner of the IDE display. Ensure the baud rate is set to 9600.

Perform the following test procedure.

Step	Action	Expected Results
1	Ensure the Gate Switch is closed and the optical lane sensors are properly illuminated (not obstructed).	N/A

2	On the serial monitor enter the letter D in the command line, press ENTER or click on the Send button.	The following message is displayed on the monitor:
		`Debug Status:`
		`- All Inputs OK`
		
		If the gate switch, reset switch or a lane sensor is found to be in the wrong state the debug status message will contain one or more of the following:
		`- Gate Switch: OPEN`
		`- Reset Switch: CLOSED`
		`- Lane 1 OBSTRUCTED`
		`- Lane 2 OBSTRUCTED`
		
		Troubleshoot and correct the problem before continuing.

3	Obstruct (block) the light illuminating the Lane 1 optical sensor. Then on the serial monitor enter the letter R in the command line, press ENTER or click on the Send button.	N/A • The red "Not Ready" LED is illuminated. • The letter "O" is displayed on the monitor indicating the track is not ready.
4	Repeat step 3 for each lane.	Same as step 3.
5	Set the Gate Switch to the open position. On the serial monitor enter the letter R in the command line, press ENTER or click on the Send button.	N/A • The red "Not Ready" LED is illuminated. • The letter "O" is displayed on the monitor indicating the track is not ready.

Step	Action	Expected Results
6	Set the Gate Switch back to the closed position.	N/A
7	Ensure the Gate Switch is closed and the optical lane sensors are properly illuminated.	N/A
8	On the serial monitor enter the letter G in the command line, press ENTER or click on the Send button.	The character "." is displayed on the monitor.
9	On the serial monitor enter the letter R in the command line, press ENTER or click on the Send button.	• The yellow "Ready" LED is illuminated. • The letter "K" is displayed on the monitor.
10	Set and hold the Gate Switch to the open position.	• The green "Racing" LED is illuminated. • The letter "B" is displayed on the monitor indicating the race is in progress.
11	Wait 10 seconds.	After 10 seconds: • The following time message is displayed: 1 - 9.9999 2 - 9.9999
12	Set the Gate Switch back to the closed position.	• The letter "K" is displayed on the monitor indicating the timer has auto-reset for the next race after detecting the gate switch and optical sensors are in the correct state to start the next race. • The yellow "Ready" LED is illuminated
13	Ensure the Gate Switch is closed and the optical lane sensors are properly illuminated.	N/A
14	Momentarily depress the RESET switch.	• The yellow "Ready" LED is illuminated. • The letter "K" is displayed on the monitor.
15	Momentarily toggle the Gate Switch to the open position and then back to the closed position.	• The green "Racing" LED is illuminated • The letter "B" is displayed on the monitor indicating the race is in progress.
16	Wave your hand over all lane sensors to obstruct the light illuminating them before the 10 second timeout has elapsed.	The moment the last lane sensor is tripped: • Times are displayed on the monitor in the following format followed by the letter "K": 1 - x.xxxx 2 - x.xxxx K *Where x.xxxx is the time in seconds.* • The yellow "Ready" LED is illuminated
17	Repeat steps 15 and 16 as desired obstructing the light to the lane sensors in different sequences.	Same results as steps 15 & 16.

Step	Action	Expected Results
18	On the serial monitor enter the letter N in the command line, press ENTER or click on the Send button.	The message “numl=2” is displayed on the monitor.
19	On the serial monitor enter the letter V in the command line, press ENTER or click on the Send button.	The message “vert=1.00” is displayed on the monitor.
20	On the serial monitor enter the letter R in the command line, press ENTER or click on the Send button.	• The yellow “Ready” LED is illuminated. • The letter “K” is displayed on the monitor.
--	TEST COMPLETE	

Troubleshooting: Use of the Arduino IDE serial monitor or another serial terminal program can be used to observe these commands or to send the reset “R” command to the Arduino. Feel free to contact the author via email at billv923@outlook.com for additional help.