Ladder Control

Contents

Revision History:

- 13 December 2022 Initial version

Jon E. Schmidt
jontenor@gmail.com

The Project

Model Railroad Control Systems designed an Arduino-based board to control relays. This device, the Arduino Relay Breakout For Arduino Nano, was originally used for controlling special effects such as lighting effects for a model railroad. The sketch for this purpose may be found at https://github.com/joneschmidt/EffectController.

This board can be used to control turnouts and other devices. This document describes an Arduino sketch for that use, "Ladder Control." This and associated documents may be found at https://github.com/joneschmidt/LadderControl.

Arduino Sketch: LadderControl

The software controls a set of turnouts (or devices) based on the activation of a pushbutton indicating the desired destination. Each destination requires an activating button, and a path to get to that destination. The user defines the tracks and associated buttons, and each turnout in the set.

- Turnouts may be stall (continuous power) or momentary.
- Momentary turnouts require an additional relay to turn on and turn off the power to the switch machine.
- Stall turnouts optionally may have an optional relay to turn on and turn off the power to the switch machine.

Not yet implemented:

- I2C connection to additional sensors/controls.
- Alternate paths.

The Design

The user defines their ladder by specifying destination tracks and turnouts into tables. The sketch will walk the tables and initially turn off power to any power relays it finds. It then goes into a loop waiting for a button to be pushed.

When a button is pushed the sketch walks the tables again and builds the path of turnouts it needs to throw to get to that path. It will confirm that the path is unoccupied before proceeding and then throws those turnouts from the selected track (bottom) up. It will only change turnouts in the selected path. It will then loop waiting for a different button to be activated to select a track.

Software: Definitions

The user must set up tables in the LadderControl sketch to define their trackage. Please refer to the Excel sheet, LadderControlExamples.xlsx for example references below. The data structures which must be built are here:

Each destination track or route requires an entry in the track table. That entry identifies the sensor port for the button for that track, the turnout name of the turnout accessing that track, and the direction of the turnout required to get to that track, i.e. normal or reverse.

```
// Structure to define destination track selection
typedef struct {
  // track select button port
  uint32_t ButtonPort;
  // turnout - button selects turnout and leg
  String TurnName;
  // turnout leg
  int TurnLeg;
} DestnDef;
```

Each turnout or device requires an entry in the turnout table. That entry identifies the name of the turnout (“`TurnName`”), the turnout motor type (“`TurnType`”, stall or momentary), the control port for the direction control of that turnout (“`TurnPort`”), the port to control turnout motor power (“`TurnPowrPort`”), the sensor port indicating occupancy of the turnout (“`TurnOccPort`”), a path weight for the turnout for alternate route selection (“`TurnWeight`”, not implemented), the name of the next turnout up the route (“`TurnUpName`”), and the direction of the next turnout required to get to this turnout track (“`TurnUpLeg`”). There are two additional fields which are currently unimplemented, TurnUpName2 and TurnUpLeg2.

```
typedef struct {
  // turnout id - text name of turnout
  String TurnName;
  // turnout type - STALL or MOMNTRY
  uint8_t TurnType;
  // turnout control port
  uint32_t TurnPort;
  // turnout power port - required for MOMNTRY
  uint32_t TurnPowrPort;
  // turnout occupied sensor port
  uint32_t TurnOccPort;
  // turnout path value
  int TurnWeight;
  // upstream turnout id - next higher turnout
  String TurnUpName;
  // upstream turnout leg - which leg does this turnout connect to
  int TurnUpLeg;
  // 2nd upstream turnout id - if an upward facing turnout,
  // which other turnout does it connect to?
  String TurnUpName2;
  // 2nd upstream turnout leg
  int TurnUpLeg2;
} TurnoutDef;
//
```

Examples

Please refer to the “Simplest” sheet in the Excel file LadderControlExamples.xlsx. Simplest shows a two track yard, with one turnout. This is only two destinations, Track 0 and Track 1. The constants below define this situation.

```
// USER: number of turnouts in ladder
#define NUMTURNOUTS 1
// USER: number of destination tracks
#define NUMTRACKS   2

Turnout definitions on row 17:
// field definitions for the turnout table
TurnName – text name of the turnout “TO1”
```

TurnType – “STALL”
TurnPort – the control port for the relay to set direction “D2”
TurnPowrPort – the control port for the power relay: required for MOMNTRY machines, optional for STALL machines – 0 – no port
TurnOccPort – turnout occupied sensor: optional; path will not change if the path is occupied – 0 – no occupied sensor
TurnWeight – not yet implemented (NYI) – for alternate path selection - 0
TurnUpName – upstream turnout name – “” for no upstream turnout
TurnUpLeg – upstream turnout leg, normal or diverge - 0
TurnUpName2 – 2nd upstream turnout for alternate path selection (NYI)
TurnUpLeg2 – 2nd upstream turnout leg for alternate path selection (NYI)

Destination Track definitions on rows 22-23:
// field definitions for the tracks table
ButtonPort – the port for the button selecting the destination track
TurnName – name of the first upstream turnout “TO1”
TurnLeg – upstream turnout leg – normal “TNORM” or reverse “TREV”

Additional examples are found in sheets StraightLadder, ComplexLadder, and Wye.

*** end ***