Ladder Control Arduino Solution

This solution uses the Universal I/O Code set to build the Ladder Control sketch for Arduino.

Revision History:

2026-01-26 J. Schmidt initial version

Universal I/O Code

The Universal I/O Code set allows a simple integrated definition of Arduino ports and subsequent reference to those ports via a universal read/write interface regardless of the port type. Port types supported:

- Arduino digital ports
- Arduino analog ports
- Arduino analog ports connected to photoresistors
- I/O Extender ports

Ladder configuration

Ladder Control sketch features:

- Support for stall or older momentary turnouts
- Optional control of power to turnout motors/solenoids
- Optional occupancy detection prevents turnout movement under trains
- Digital or directly-connected photoresistor inputs

Step 1: Define the code options, ladder and table sizes

```
// vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
// vvvvvvvvvvvv if using the IOX and wire librarys define USEIOX true
#define USEIOX true
// ^^^^^^^^^^^^ if using the IOX and wire librarys define USEIOX true
// vvvvvvvvvvvv if using photosensors define PHOTOSENSE true
#define PHOTOSENSE true
// ^^^^^^^^^^^^ if using photosensors define PHOTOSENSE true
```

Example:

```
// DEFINE THE LADDER
// Example - ladder
//               |
//       _______/*\____Trn00 - Sns00*
//       |    N       R|
//       |             | Tracks
//       |             |
//       |          __/*\_____Trn01 - Sns01*
//       |         N       R  |
//       |         |          |
//       |      __/ \__Trn02  |
//       |     | N     | R    |
//    TBut00 TBut01 TBut02  TBut03


// vvvvvvvvvvv DEFINE THE NUMBER OF PORTS
// USER: number of Arduino & XIO ports used
#define NUMPORTS    11
// USER: number of turnouts in ladder
#define NUMTURNOUTS 3
// USER: number of tracks
#define NUMTRACKS   4
```

Step 2: Configure the I/O ports

Each port is configured by

- The *handle:* #define to be used by later definitions
- The *PortAddr* which is the physical address of the Arduino or IOX port
- The *PortIO* which defines the port type:
    - INPUT
    - INPUT_PULLUP
    - OUTPUT
    - INPUT_PHOTO for photoresistor inputs

```
// define the Arduino ports
// buttons, sensors, and turnouts
// #define is the HANDLE -> position in the table
PortDef const PortS [NUMPORTS] = {
//  PortAddr - PortIO
// Buttons
#define But00P 0
       XADR(0x24,0), INPUT_PULLUP, // 0
#define But01P 1
       XADR(0x24,1), INPUT_PULLUP, // 1
#define But02P 2
       6, INPUT_PULLUP, // 2
#define But03P 3
       8, INPUT_PULLUP, // 3
// Turnouts
#define Trn00P 4
       XADR(0x24,5), OUTPUT,       // 4
#define Trn01P 5
       7, OUTPUT,       // 5
#define Trn02P 6
       9, OUTPUT,       // 6
// Turnout power
#define TPwr00P 7
   10, OUTPUT,       // 7
// Sensors
#define Sns00P  8
       A2, INPUT_PHOTO, // 8
#define Sns01P  9
       A3, INPUT_PHOTO, // 9
#define StatLED  10
       13, OUTPUT, // 10
};
```

Note: Extended IO ports are defined by the macro XADR({board},{bit}). For example, XADR(0x24,0) is XIO board 24x and bit 0. The sketch controls the boards such that each bit is configured separately, either input or output.

Step 3: Configure the turnouts

Each turnout is configured by:

- The *__handle__*: #define to be used by later definitions
- The *__TurnPortHandle__*: *Handle* to the Arduino port
- The *TurnTurnType:* STALL or MOMNTRY
- The *__TurnPowrHandle__*: *Handle* to the Arduino port for controlling power to the turnout(s) or NONE
- The *__TurnOccHandle__*: *Handle* to the Arduino port for detecting occupancy or NONE
- The *__TurnDUpHandle__*: *Handle* to the upstream turnout or NONE
- The *__TurnUpLeg__*: Upstream turnout leg normal or reverse: TNORM - TREV

Example:

```
// #define is the HANDLE -> position in the table
// TurnPortHandle, TurnType, TurnPowrHandle, TurnOccHandle, TurnDUpHandle TurnUpLeg
#define Trn00D 0
    Trn00P,     STALL,     NONE,           Sns00P,       NONE,         NONE, // 0 top
#define Trn01D 1
    Trn01P,     MOMNTRY,   TPwr00P,        Sns01P,       Trn00D,       TREV, // 1
#define Trn02D 2
    Trn02P,     STALL,     NONE,           NONE,         Trn01D,       TNORM // 2
```

Step 4: Configure the buttons

Each button controls the route to a track:

- The *ButtonHandle: Handle* to the Arduino port for the input signal
- The *TurnDHandle: Handle* to the turnout
- The TurnLeg: Turnout leg normal or reverse: TNORM - TREV

```
TrkButnDef const TrkButnS [NUMTRACKS] = {
// ButtonHandle  TurnDHandle   TurnLeg
// track 0
      But00P,     Trn00D,      TNORM,
// track 1
      But01P,     Trn02D,      TNORM,
// track 2
      But02P,     Trn02D,      TREV,
// track 3
      But03P,     Trn01D,      TREV
  }; // TrkButnS
```

Actions

When the sketch starts it will validate the configuration looking for problems with the port, turnout, and button definitions. If a StatLED is defined, a rapid blinking will indicate an error or configuration problem.

The sketch will then wait for an input to be triggered. The StatLED will be set to HIGH. The sketch will then go through the path indicated and check if the path is occupied. If occupied it will flash rapidly and ignore the trigger. Otherwise it will align the path as requested.