EzyBus and a turntable.

I am a member of both the West and East groups and when Davy Dick was looking for ‘volunteers’ to test build the first production EzyBus system the East group decided I was a suitable ‘volunteer’ because I am good with a soldering iron !

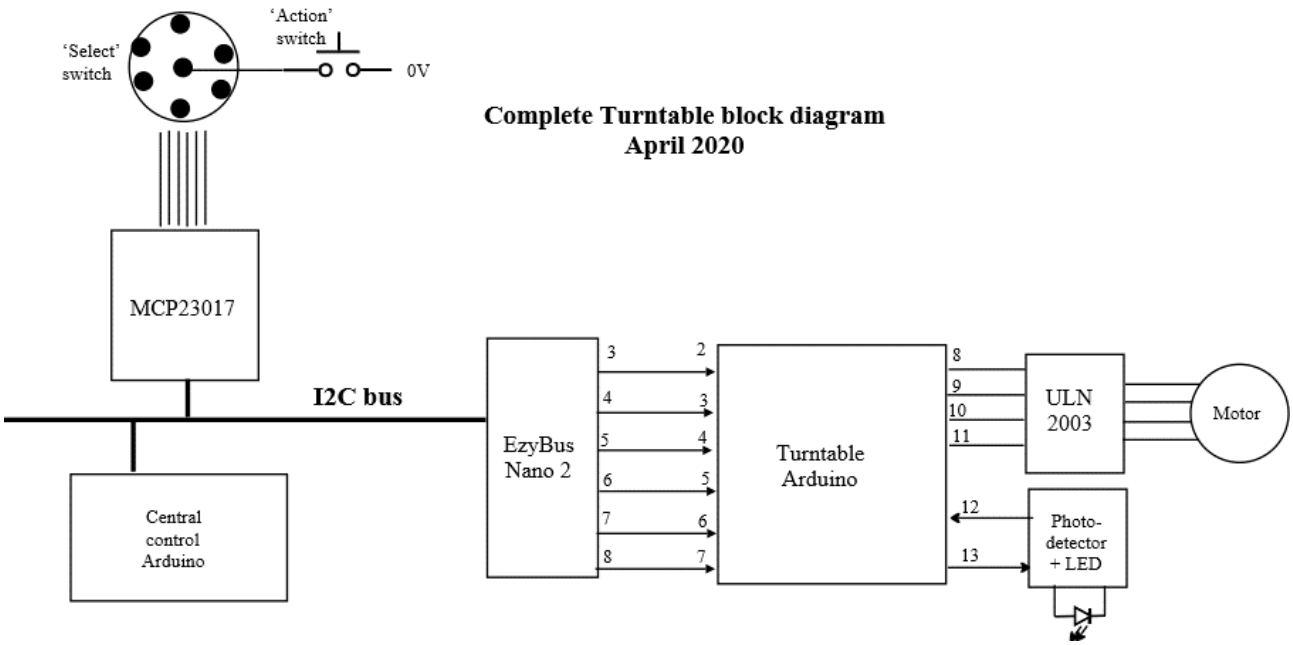
My background to all this is that I once trained as an electronics service tech on audio and TV systems so I have some theoretical knowledge and I later did an HNC in Mechatronics which is where I learnt about computers. I am not a railway modeller but back in the early 90’s I designed and built the control system for a friend’s garden railway and some of that equipment is still in use. I also have virtually no formal training in programming ! But once my Eastern colleagues saw my breadboard they decided to build a demonstration layout to highlight EzyBus.

After much discussion a layout was designed. A single track N-gauge extended oval with on one side two passing tracks with a spur off these to a turntable with four arms: the entry, a shed track and two parking tracks, and, on the other side of the oval a level crossing. The first challenge was the level crossing and that is basically resolved although the sensors need to be tested with a real train on a piece of real track. So, on to the turntable . . .

What did I know about Arduinos and turntables ? Not a lot ! So the starting points were what the group wanted it to do and a literature search. Plug ‘Arduino + turntable’ into the search engine of your choice and an impressive amount of information turns up. The Arduino forums, various modelling forums and general hobby engineering sites. There is also an amazing amount of code published most of which you are free to use and modify.

The hardware.
A member donated a Peco turntable. At least as a starting point it was decided that a cheap and cheerful 28BYJ-48 stepper motor and its driver, the ULN2003, would be used controlled by a Uno and using a PMP 22 Laser TOTI as the zero point detector. It is a characteristic of stepper motors that they have no memory, they need to be zeroed at start up so that the motor knows where it is as you control them by telling them how many steps to take in which direction You also need to know how many steps it takes for a complete revolution. This particular motor takes 64 steps for it to rotate once but it comes with a built in gearbox. These are not all the same and this leads to much debate. The only answer is to suck it and see. Write a program to make it rotate say an intended five turns of the turntable and see if it does. If it doesn’t, plug in a different value for steps per rotation until you get it right. In the case of the motor I am using it takes 2040 steps for a complete revolution of the turn table. One nice thing about Arduinos is they are amazingly easy to re-program.

Control
The turntable will be controlled by a six position rotary switch and triggered by a push-button switch to ground. The six positions feed into an MCP23017, the input point for the EzyBus system. And from there onto the I2C bus. If any of these inputs is triggered the controlling Uno detects them and passes the appropriate signals to the output switching Nano, again along the I2C bus. Six digital outputs from the Nano are used to feed the turntable Arduino to control the stepper motor. A variation in the hardware is that the Laser TOTI has been replaced with a photo detector. It was found that alignment of the laser, the reflector and the photo-transistor was overly critical and went out of alignment too easily. Also, because of the very narrow ‘window’ in the detector, zeroing became much more accurate and consistent This is the complete block diagram -



The program.
As you will see from the code intro this has gone through many iterations being currently on v9C, not counting off-shoots, as I fumbled my way towards what I believe is essentially the final version. It could benefit from being tidied up but it works and meets the objectives, which are -

1/ To rotate from any one position to any other by the shortest route
2/ To rotate towards the entry track from any position in either direction to allow locos to change direction of travel on the system
3/ To allow 180^{o} turns from any track

A 20 second delay is built in to allow the EzyBus system to settle before it activates and the zeroing process is done in void setup(). Pin13 (output) turns on the photo-detector and while pin 12 (input) is low the turntable rotates until it interrupts the photo-detector. At which point the turntable pauses for 1 second then rotates on automatically to line up with the entry track. Then control passes to void loop() and the turntable responds to the 'Select' switch.

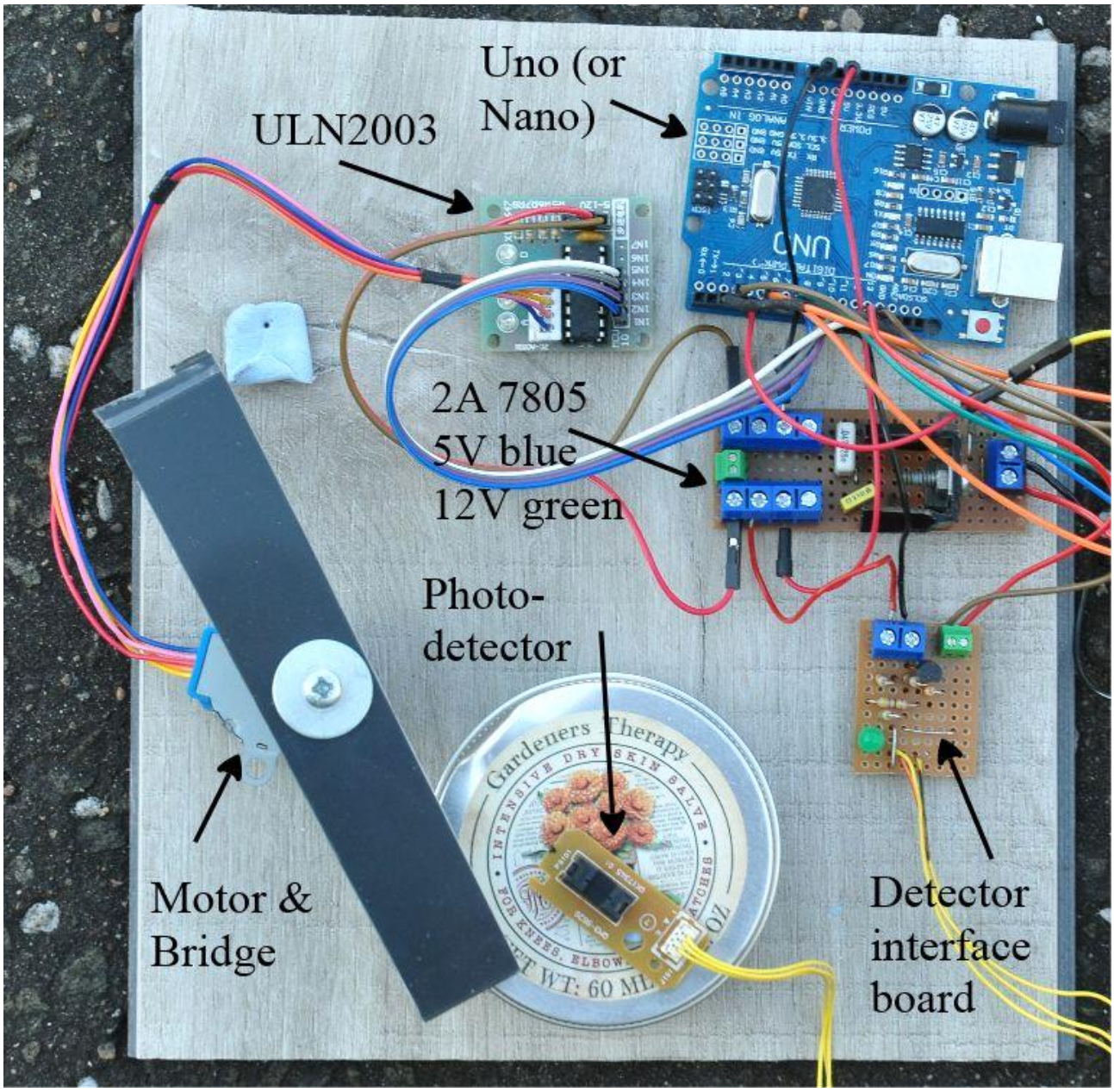
The values for stepCount between tracks in the program are purely for test and development purposes and 'real' ones ones would need to be entered in their place. To this end I created a very quick and dirty counting program which works and if anybody wants it I will supply the code. While counting I use a data only cable as it is important to disconnect the power supply to the turntable Arduino when programming it. If you don't disconnect power it will try to drive the complete system through the power coming down the USB cable. This is not good for either your computer or the Arduino !

This program will work equally well, as written, on either aa Uno or a Nano but is for bi-polar stepper motors. If a uni-polar motor was to be used it would need a substantial re-write, particularly with regard to pin definitions and void setup() as the methods of controlling them are quite different.

Problems.
Getting to grips with the Arduino and stepper motors has been a steep learning process for me as I had never played with either before this project started. Apart from that the biggest problem has been project management, or rather lack of it. Because some parts of this project were either simpler or fell well within various member's skills and resources they were decided on and finished before

other parts and some of these parts impinged upon later thoughts and introduced some constraints and frictions. It is also important to spell out system requirements in some detail before starting to write a program. This didn't always happen.

The breadboard lay-out.



Chris Cosgrove
trainscpc@aol.com
April 2020

Click HERE to see video

The code.

```
// DetectorTest_9C.ino
//
// A program to control a model railway turntable with four tracks
// accessing it. It uses a photo-detector as the zero point detector and
// it should go from any one track to any other. The hardware used
// is a Uno and a ULN2003 controlled 28BYJ-48 uni-polar stepper motor.
// A bi-polar stepper could be used but would require a considerable
// re-write.
//
// Arduino IDE 1.8.12
//
// Chris Cosgrove, April 2020
//

#include <Stepper.h>

//  Motor pin definitions
#define mPin1  8     // IN1 on the ULN2003 driver 1
#define mPin2  9     // IN2 on the ULN2003 driver 1
#define mPin3  10    // IN3 on the ULN2003 driver 1
#define mPin4  11    // IN4 on the ULN2003 driver 1

//  Other pin definitions
#define eb1  3       //  Is Switch Posn 4t, allow 180 rotations
#define eb2  4       //  Input 2 from EzyBus, go to Track 1, Main
#define eb3  5       //  Input 3 from EzyBus, go to Track 2, Shed
#define eb4  6       //  Input 4 from EzyBus, go to Track 3, Bay 1
#define eb5  7       //  Input 5 from EzyBus, go to Track 4, Bay 2
#define eb6  2       //  Is Switch Posn 2T, rotate to Track 1, long way
#define photo_d  13    //  Output to detector
#define detector  12   //  Input from detector

const int stepsPerRevolution = 2040;
int stepCount;              //  number of steps the motor has taken
int photo_d_state;          // int__states added on advice from
int detector_state;         // Davy Dick to hold values read in from
int eb1_state;              // pins 2 - 7 and 11,12
int eb2_state;
int eb3_state;
int eb4_state;
int eb5_state;
int eb6_state;
int current_track;
int old_current_track;

// initialize the stepper library on pins 8 through 11:

Stepper stepper1(stepsPerRevolution, mPin1, mPin3, mPin2, mPin4);

void setup() {                      // Start of 'void setup'.
```

```
 pinMode(photo_d, OUTPUT);
 pinMode(detector, INPUT);
 pinMode(eb1, INPUT);
 pinMode(eb2, INPUT);
 pinMode(eb3, INPUT);
 pinMode(eb4, INPUT);
 pinMode(eb5, INPUT);
 pinMode(eb6, INPUT);

 delay(20000);                    // 20 second delay to allow Ezybus to initialise
 digitalWrite(photo_d, HIGH);
 delay(5);                        // To allow detector to settle
 detector_state = digitalRead(detector);

 while (detector_state == LOW)            // While detector output is LOW move on one
 {                                // step and check detector again.
  stepper1.step(1);
  detector_state = digitalRead(detector);
  (stepCount++);
  delay(10);
 }

 if (detector_state == HIGH)              // laser detected
 {
  stepper1.step(0);                       // End movement
  delay(5);
  digitalWrite (photo_d, LOW);
  delay(1000);
  stepper1.setSpeed(2);                   // Then move to main entry
  stepper1.step(200);
  delay(100);
  current_track = 1;
  stepCount = 0;                          // Set Zero point, Main
  }
}                                  // End of 'void setup'.


 void loop() {                     // Start of 'void loop'

 /* There are 4 tracks, 1 to 4 in order Main, Shed, Bay 1, Bay 2. On zeroing the TT
  * goes automatically to the Main position. All that is required is that the number
  * of steps between the zeroing point and Track 1 is known. For test it is assumed
  * to be 200. The other test spacings are -
  * Main to shed = 450 steps;    Shed to Bay 1 = 810 steps
  * Bay 1 to Bay 2 = 430 steps;  Bay 2 to Main = 350 steps
  * For clockwise movements steps to move are as above. For anticlockwise 10 steps
  * are added then the TT is driven 10 steps clockwise so that, for reasons of gear
  * backlash. the TT is always stopped from the same direction.
 */

  eb1_state = digitalRead(eb1);
```

```
 eb2_state = digitalRead(eb2);
 eb3_state = digitalRead(eb3);
 eb4_state = digitalRead(eb4);
 eb5_state = digitalRead(eb5);
 eb6_state = digitalRead(eb6);

if (eb2_state == HIGH && current_track == 2)       // Move to Track 1,
{
  stepper1.setSpeed(2);
  stepper1.step(-460);
  delay(100);
  stepper1.step(10);
  current_track = 1;
}

if (eb2_state == HIGH && current_track == 3)       // Move to Track 1
{
  stepper1.setSpeed(2);
  stepper1.step(780);
  current_track = 1;
}

if (eb2_state == HIGH && current_track == 4)       // Move to Track 1
{
  stepper1.setSpeed(2);
  stepper1.step(350);
  current_track = 1;
}

if (eb3_state == HIGH && current_track == 1)       // Move to Track 2
{
  stepper1.setSpeed(2);
  stepper1.step(450);
  current_track = 2;
}

if (eb3_state == HIGH && current_track == 3)       // Move to Track 2
{
  stepper1.setSpeed(2);
  stepper1.step(-820);
  delay(100);
  stepper1.step(10);
  current_track = 2;
}

if (eb3_state == HIGH && current_track == 4)       // Move to Track 2
{
  stepper1.setSpeed(2);
  stepper1.step(800);
  current_track = 2;
}
```

```
if (eb4_state == HIGH && current_track == 1)        // Move to Track 3
{
  stepper1.setSpeed(2);
  stepper1.step(-790);
  delay(100);
  stepper1.step(10);
  current_track = 3;
}

if (eb4_state == HIGH && current_track == 2)        // Move to Track 3
{
  stepper1.setSpeed(2);
  stepper1.step(810);
  current_track = 3;
}

if (eb4_state == HIGH && current_track == 4)        // Move to Track 3
{
  stepper1.setSpeed(2);
  stepper1.step(-440);
  delay(100);
  stepper1.step(10);
  current_track = 3;
}

if(eb5_state == HIGH && current_track == 1)         // Move to Track 4
{
  stepper1.setSpeed(2);
  stepper1.step(-360);
  delay(100);
  stepper1.step(10);
  current_track = 4;
}

if(eb5_state == HIGH && current_track == 2)         // Move to Track 4
{
 stepper1.setSpeed(2);
 stepper1.step(-810);
 delay(100);
 stepper1.step(10);
 current_track = 4;
}

if(eb5_state == HIGH && current_track == 3)        // Move toTrack 4
{
  stepper1.setSpeed(2);
  stepper1.step(430);
  current_track = 4;
}

if(eb1_state == HIGH)                   // This allows all tracks to rotate 180
{
```

```
  old_current_track = current_track;
  stepper1.setSpeed(2);
  stepper1.step(1020);
  current_track = old_current_track;
 }

 if(eb6_state == HIGH && current_track == 2)  // Alternate rotations to Track 1, Main entry
 {
  stepper1.setSpeed(2);
  stepper1.step(570);
  delay(10);
  current_track = 1;
 }

 if(eb6_state == HIGH && current_track == 3)
 {
  stepper1.setSpeed(2);
  stepper1.step(-250);
  delay(100);
  stepper1.step(10);
  current_track = 1;
 }

 if(eb6_state == HIGH && current_track == 4)
 {
  stepper1.setSpeed(2);
  stepper1.step(-680);
  delay(100);
  stepper1.step(10);
  current_track = 1;
 }


}                                   // End of void loop()
```