June 5, 2015 at 23:02

1. Introduction. This is the firmware portion of the propulsion system, featuring piruett turning.

This will facilitate motion by taking "thrust" and "radius" pulse-width inputs and converting them to the appropriate motor actions.

Both pulse-width inputs will have some dead-band to allow for full stop.

The pulse-width from the receiver will probabily be at 20 ms intervals. The time will range from 1000–2000 ms, with 1500 ms being for stopped. That will need to be measured.

Port motor pulse will be applied to ???, starboard will be at ???. They will be sampled at about 1000 times per second. The median time will be subtracted from them for a pair of signed values thrust and yaw. The value will be scaled.

The sum and difference of thrust and yaw will be translated to power to the port and starboard motors. When near median the motors will be disabled. The motors will also be disabled when there are no input pulses. Each motor need direction and power so that's 4 signals of output. Afdding the two signal of input, I need more I/O than the trinket has. So—I put an order in for a Pro Trinket with far more capability. It has an ATmega328.

Jaw and fire control could be added to this board too. We will see.

The ATmega328 has a fancy 16 bit PWM with two comparators, Timer 1. This will do more than fine for the two motors.

place-holder code below ==========

Extensive use was made of the datasheet, Atmel "Atmel ATtiny25, ATtiny45, ATtiny85 Datasheet" Rev. 2586QAVR08/2013 (Tue 06 Aug 2013 03:19:12 PM EDT).

⟨ Include 4 ⟩
⟨ Types 5 ⟩
⟨ Prototypes 6 ⟩

2. `"F_CPU"` is used to convey the Trinket clock rate.

```
#define F_CPU 16000000UL
```

3. Here are some Boolean definitions that are used.

```
#define ON 1
#define OFF 0
#define SET 1
#define CLEAR 0
```

4. ⟨ Include 4 ⟩ ≡

```
#include <avr/io.h>      /* need some port access */
#include <util/delay.h>      /* need to delay */
#include <avr/interrupt.h>      /* have need of an interrupt */
#include <avr/sleep.h>      /* have need of sleep */
#include <stdlib.h>
#include <stdint.h>
```

This code is used in section 1.

5. Here is a structure to keep track of the state of things.

```
⟨ Types 5 ⟩ ≡
  typedef struct {
    uint8_t wavecount;      /* delay to remain as if waveless, to ensure waves */
    uint16_t armwait;       /* countdown index to arm siren */
    uint8_t armed;      /* non-zero indicates that the siren is armed */
    const uint8_t nowavecount;      /* time until siren arm */
  } statestruct;
```

This code is used in section 1.

6. ⟨ Prototypes 6 ⟩ ≡

```
  void ledcntl(uint8_t state);      /* LED ON and LED OFF */
```

This code is used in section 1.

7. Here is *main*( ).

```
  int main(void)
  {
    ⟨ Initialize pin outputs and inputs 11 ⟩ledcntl(OFF);
    return 0;      /* it's the right thing to do! */
  }      /* end main() */
```

8. Here is a simple function to flip the LED on or off.

```
  void ledcntl(uint8_t state)
  {
    PORTB = state ? PORTB | (1 ≪ PORTB5) : PORTB & ∼(1 ≪ PORTB5);
  }
```

9.

10. These are the supporting routines, procedures and configuration blocks.
Here is the block that sets-up the digital I/O pins.

11. ⟨ Initialize pin outputs and inputs 11 ⟩ ≡

```
{   /* set the led port direction; This is pin #13 */
  DDRB |= (1 << DDB5);
}
```

This code is used in section 7.

⟨ Include 4 ⟩ Used in section 1.
⟨ Initialize pin outputs and inputs 11 ⟩ Used in section 7.
⟨ Prototypes 6 ⟩ Used in section 1.
⟨ Types 5 ⟩ Used in section 1.