Contents

1 Attiny Solar Energy Harvest Tests

I have the following:

- Solar panels
- Attiny 10

To this list, I will add a supercap, and maybe an energy harvesting IC. The goal being to load the super cap during the day, and to run 24/7. I will need an exceptionally low power micro. The super cap will need to be about 3.3V or 5V.

1.1 Micro Considerations

The Arduino Atmega328P is not an option. I'm looking to have a current draw of only 1mA max, (ideally 500uA) when active. Moteino is also not an option for this. Those are made for batteries. I want to be battery free [1]. A super cap, however can be used to store energy. I'll get to that shortly.

For micros, I have some Attiny10 on hand, and these have a reasonably low power pull in active mode. Let's build those up first. What will the micro do? Not sure yet.

1.1.1 Micro Notes

Must run at 1.8V / 1MHz per front page of data sheet, for 200uA draw in active mode.

[1] I might backtrack on this, but I think I'll do one with and one without batteries.

1.1.2 Energy Storage

I don't want a battery. Let's go with a super cap. The solar panels will only be active some of the time, so I will want to harvest energy with some kind of IC into the cap when the sun is out.[2]

1.1.3 Make parts, not scrap

I want to make sure that all parts I build are perf board parts, not breadboard scrap (to be torn down and rebuilt again). This is an Attiny, so no need to test much, yet. Breadboards never really worked for me, so far. Too much build up, then tear down... I like things that are built to last. Parts, not scrap.

1.2 Programming

To program the Attiny10, I'll use the Arduino adapter from the Junk + Arduino blog. I built it up[3], and was able to read the memory. In order to upload to the board, you will need a compiler setup. You can possibly do it in AVRGCC, but instead I opted for either Arduino IDE (via Attiny10Core which didn't work), and then went to Mplab. In order for mplab 5.25 to work, it will need XC8 compiler, and there is a pack that can be downloaded through the IDE to get Attiny10 support. After that you can get .hex files to upload.

It appears the AVR Dragon (which I have) can not be used. However, other programmers can be used. Pickit 4, Mkavrii, stk600, I think.

[2]Reference: www.analog.com/media/en/technical-documentation/technical-articles/solarenergyharvesting.pdf is a start. I'll need to do more research.

[3]Had slight error where the Arduino + board wouldn't read - pins too short on headers, then the arduino wouldn't boot - due to bad connection on perf board shield. Thankfully, the USB port didn't try to run. Protection circuitry cut in on the laptop. This was all on a perf board. My proto pcb doesn't have this issue.

1.2.1 Testing Arduino Loader

Tested this with the blink_LED.c in code folder. The code is as simple as possible. It is the following:

```
1  //#include <xc.h>
2  #include <avr/io.h>
3  #include <util/delay.h>
4
5  int main(void)
6  {
7      // PB2 output
8      DDRB = 1<<2;
9
10     while(1)
11     {
12         // Toggle PB2
13         PINB = 1<<2;
14         _delay_ms(500);
15     }
16 }
```

[4] When programmed in Mplab, with XC8 compiler, and Attiny10 support, I get the following hex output:

```
:100000000AC020C01FC01EC01DC01CC01BC01AC01B
:1000100019C018C017C011271FBFCFE5D0E0DEBF41
:0A002000CDBF03D000C0F894FFCF5D
:10002A0044E041B940B95FE966E871E05150604087
:0A003A007040E1F700C00000F5CFB0
:02004400DDCF0E
:00000001FF
```

The content of this hex isn't the focus of this passage. Instead, I want you to review the results of a D for Dump Memory, by the Arduino Loader.

[4] As a recap, this code is setting the 2nd bit of DDRB register high (1), and also the PINB register, bit 2. From this we get blinky, aka a square wave.

```
Current memory state:

registers, SRAM
      +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
0000: 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030: 00 00 00 00 00 00 03 00 00 79 00 03 00 00 00 00
0040: B7 AD AE FA 58 70 63 6B FB 5A B4 1B FF FF 35 3F
0050: 67 D7 33 43 DF 5F FB 72 C9 7D FE E9 9D C5 00 12
NVM lock
      +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
3F00: FF FF
configuration
      +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
3F40: FF FF
calibration
      +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
3F80: 79 FF
device ID
      +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
3FC0: 1E 90 03 FF
program
      +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
4000: 0A C0 20 C0 1F C0 1E C0 1D C0 1C C0 1B C0 1A C0
4010: 19 C0 18 C0 17 C0 11 27 1F BF CF E5 D0 E0 DE BF
4020: CD BF 03 D0 00 C0 F8 94 FF CF 44 E0 41 B9 40 B9
4030: 5F E9 66 E8 71 E0 51 50 60 40 70 40 E1 F7 00 C0
4040: 00 00 F5 CF DD CF FF FF FF FF FF FF FF FF FF FF
4050: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
4060: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
(...some memory omitted here for brevity...)
43E0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
43F0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Notice that the "AC020C01F" is set. That is from the hex. But

the 01000... before it seems to be missing. Some deciphering of how the Arduino programs the Attiny is in order here. It also doesn't end the same.

Regardless, when programming, the Arduino reports 70 bytes written, and in the Mplab project memorymap.xml file, it notes 70 bytes for the sketch. This lines up.[5]

The blinking LED works. Let's move on.

1.2.2 Conclusion on Arduino Programming Attiny10

It's possible, but you have to make a dedicated jig (almost), so it might be easier to use the official programming tools. However, based on this https://www.avrfreaks.net/forum/pickit-4-and-avr-mcu I might not have a choice. So I will use the Arduino for now. But will have to devise what on board parts are req'd for programming, and incorporate into proto board layout. **NOTE: This was before I build the proto pcb.**

1.2.3 IO Port Switching Speed

Using the above code without any delay_ms, I get the following results from a default clock speed, and a 128KHz clock speed. This test was done to confirm that I could change the clock with

```
1       //Write CCP
2       CCP = 0xD8;
3       //change CLK to 128KHz
4       CLKMSR = 0b01;
```

There was no issue.

```
Default CLK (8MHz? or 1MHz?): 160KHz square wave
128KHz CLK: 2.5KHz square wave on GPIO
```

I am going to pursue 128KHz for starters, for lower current dissipation. Note that with the Arduino loader, it is cumbersome to test and

[5] Although for an unknown reason, every command registers twice on the Arduino serial monitor, but this appears to be harmless.

change code as you move along. It is therefore going to be necessary to use a programmer, with a dedicated header on board. **NOTE: this was before the proto board, which made programming slightly faster. Still not 100%, but usable in terms of agility.**

1.2.4 VCC 1.8V

The lowest power supported: 1.8V can be applied, without any configuration needed. It does not affect IO switching speed (although obviously amplitude is affected).

```
128KHz CLK (5.0V): 2.5256  KHz IO switch
128KHz CLK (3.3V): 2.5477  KHz IO switch
128KHz CLK (1.8V): 2.5849  KHz IO switch
```

As voltage drops, IO speed increases.

VCC Dropout voltage:
From 1.5, it drops out at 1.248V or so. Comes back at about 1.34V

Test size of 1.

Can't run this with one (AA) battery, but you could with 2.

Current Draw: 128KHz - IO test, 1.8V, 0.08mA (78uA) (tested w/ HP 3478A)

1.3 Application

First, I need a board for these and a programmer, to quickly program. Second, I need an application. I want extremely low power. Hopefully, solar with no batteries, to start. This is extremely low - that is the point. Let's keep this ridiculous.

Given the power requirements put me under 1mA (with my current panels), I'm considering the following: EEPROMs would require

SPI protocol. Doable, but overcomplicated for now.

Eink (need to find a small and cheap enough option. So far, they have either too many pins, and/or use too much current. Something like what stores use to display prices would work, but that doesn't get the data out, only makes it readable.),

Third option would be RF. That is a viable path, but not today. Let's skip that for now.

Fourth option that comes to mind is IR. IR diodes, as in TV remotes, would work well here. I am choosing this as the first project. I will have dumb clients, that consist of - Attiny / IR / Sensor powered by solar. I will have a BBB that receives the IR data, and does all intelligent data gathering. To keep things simple, the IR will be binary ADC data, or otherwise sensor numbers. No SPI, no protocol complexity. That would require space on the Attiny.

Let's build some boards based on the above.

For sensors: While building, I came across an option. Hall effect sensors. I think also capacitive sensors can be used. This may find a use in a gate sensor, for when a driveway gate is opened or closed. With a small battery, it would work for years.

Footprints: I had to make a footprint for this module on board package for one sensor. The solution to get footprints right? copy graphic image and make it into silkscreen on the board. Easy.

The sensor I looked at was a temp and humidity sensor SHT11 (SHT10 is obsolete). It is low power enough. However, it's $20. So not in my price range. Otherwise, it would work here. Looks like communication is a shift register, or SPI.

1.3.1 Magnetic Current Sensor

There is this: BM14270AMUV-LB Which is low enough current here (<1mA). But $7 in qty, and req's I2C. Not today.

1.3.2 Accelerometers

These are an option.

Best pinout (for deadbug) is LIS344ALHTR (but lacks vcc down to 1.8) 2nd Best pinout with full 1.8 -3.6 vcc is ADXL337BCPZ-RL7 (Keep in mind, these are low end options only) (Analog output only. keep it simple for now.) Runner up to all above, is KXTC9-2050-FR . But has worse pinout.

Going with AD part. $5 in single qty. Digital output Accelmeters are cheaper.

All have tiny package sizes.

Since I am grabbing 1 output only, will need to orient or choose correct output.

PINOUT: When I said best pinout, I meant that you can solder this by hand or with hot air, without much difficulty, because the layout does not require all pins to be connected (hoping I don't get bit by floating pins, though). Or, the layout has PWR and GND together, which means some pins can be bridged.

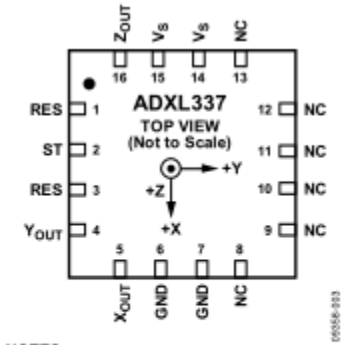

Figure 2. Pin Configuration

Figure 1: Some pinouts are easier than others. That said, this would still benefit from a custom breakout board. Bridge the V+ and GND, omit NC, RES pins and then you will want hot air and solder paste. Those looking for trouble can try magnet wire in a pinch.

1.3.3 Temperature Sensors

Temperature can be boring, but why not. Let's throw one of these on: LMT84LP . Pin compatible with LM35. Supply current is maybe 8uA. Extremely low. LM84 (1.5V starts, to 5.5), LM85 (1.8V to 5.5)

1.3.4 Gas Sensors

Lowest is 5mV as of writing on dkey. Skipping. The SHT would work, but its too expensive.

1.3.5 Supercap

For now, trying whatever I have in my junk box, as super caps are expensive. If I had money, I'd try: FG0V155ZF

1.3.6 Hall Effect Sensors - Push Pull vs Open Drain Outputs

```
9.1.1OutputTypeTradeoffs
The push-pulloutputallowsfor the lowestsystempowerconsumption
becausethereis no currentleakagepathwhenthe outputdriveshighor
 low. The open-drainoutputinvolvesa leakagepaththroughthe
 externalpullupresistorwhenthe outputdriveslow.The
 open-drainoutputsof multipledevicescan be tied
 togetherto forma logicalAND.In this setup,if any
sensordriveslow, the voltageon the sharednodebecomeslow.
 Thiscan allowa singleGPIOto measurean arrayofsensors
```

From DRV5032 data sheet.

1.3.7 Conclusion: Starting Sensors

So as a recap, to start with affordable, low power sensors for my project, I have the following types:

- Temp sensor (cheapest)
- Magnetic Sensor (hall effect)
- Movement Sensor (accellerometer) (analog output) (tiny package)
- capacitive sensor (azoteq)(may only be short range)

Output, I have not determined yet. IR will not work, as its too high power. Unless I dedicate a battery just for the IR diode... Or make it battery powered. I'll start with batteries, but for solar panel and supercap, it will likely not be viable, unless I transmit extremely rarely. That is also an option, however.

1.3.8 Farad to mA

1.5F supercap can supply 1.5A for 1 second. That is 0.025A for 60 seconds, or 25mA for a minute. Let's stay I use half that, so 12mA for 60 seconds is my supply. If I transmit once every other hour (when in sunlight)...[6]

[6]However, upon my testing, I found that the super cap is only useful while there is light. After dark, it won't hold for very long. Perhaps 1 hour.

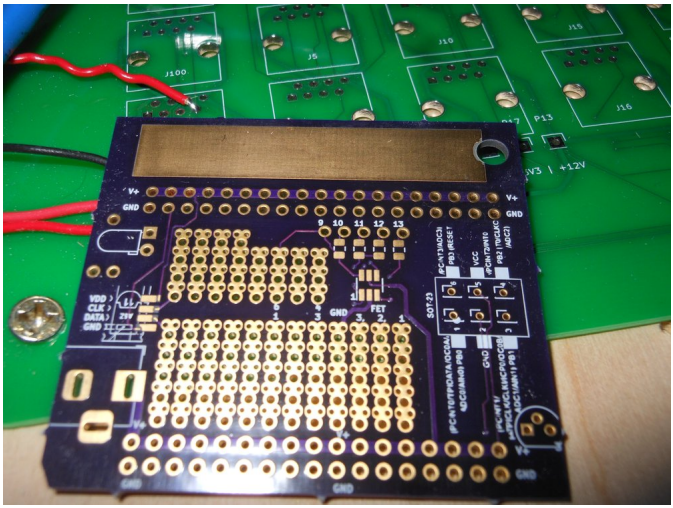

Figure 2: The silk screen pinout and instructions help.

1.4 PCB

I have built rev 2 of the board today. Using an Uno, user must remember to include VCC and GND. So the programming takes up all 6 pins. The 0.1" headers were slightly close to the resistors, and the top row of VCC and GND headers are separated, so I labeled them V+2, and GND2.[7]

The 0.05" pin headers work perfect for scope probes. The extra breakout I made demonstrates this.

1.4.1 PCB Programming

By default, the Attiny10 idles around 1.5mA (5V), before programming.

First thing to do, is to program it into a low power mode.

In order to use low power, whilst using the Uno, I will need to add a jumper to the PCB, so that you can switch between V+2, and V+1. Three pin jumper. Outside pins are each V+, internal goes to

[7]Forgetting to plug in GND, and or 5v+ (for Uno), while still plugging in the TPI pins, did not break the Attiny10, in my tests.

Attiny V+. I'll need to remove the trace on gerber rev2, that goes from 4 to VCC.

1.4.2 PCB programming and use

It's not possible to leave the Arduino plugged in, while testing the Attiny10. example: Even if you power down the Uno, and use a jumper to change V+ rails, the power dissipation through the TPI pins (10,11,12,13) will cause a draw of about 545uA, and the Attiny10 will not toggle its led. Therefore, not only a power jumper is required to use the Attiny, but also a 4 or 5 pin 0.1" cable for the TPI pins, must be disconnected before using the Attiny. [8]

1.4.3 PCB Rev3

- Added jumper to switch between different power rails, so that Arduino can be left sort of plugged in (turns out, it is still required to disconnect 9,10,11,12,13 pins, so use a 4/5 pin cable). This makes testing low power (2-3V) code, easier, while still programming with the 5V Uno.
- Added SOT23-6 breakout
- Moved Resistors further from 9,10,11,12,13 pins.
- Added separate board for analog Accelerometer.
- Moved VCC breakout (0.5" header pins) to connect directly to tiny, not to bottom Voltage rail.
- Added note about VCC and GND must be connected when programming.
- Added accelerometer pcb breakout. Package is small, but thanks to pinout/unused pins - is reasonable to solder with hot air. I knew this when I chose it... Then I forgot it earlier today.

[8] In code section, mplab - tests2, the power draw of the 128KHz internal CLK with PB2 flipping at about 400us a cycle, at 3V VCC is about 115uA. At 2V it is 95uA.

1.4.4 RF Comms

I want to use RF to communicate with this device. The IR would work, but would require batteries. If I'm going to use batteries I may as well use RF. Ideally, RF without batteries would be nice.

I searched for LoRa modules, and came across this

https://www.disk91.com/2015/technology/ networks/first-step-in-lora-land-microchip-rn2483-test/

I've added the datasheets. The RN2483 can be set with a UART, which means two pins on the Attiny. I will want to use a transistor to turn off the module (not using sleep mode), when not in use, so that's another pin. The 4th pin would be for whatever sensor I'm using. Right now, I'm thinking an accelerometer to watch for motion on a door or gate. I may need more pins, as I'd like to be able to switch the accel ic on and off as well.

There is the Attiny40, and also an Attiny402. Different, but both worth considering. The Attiny40 is covered by my programmer, the 402 is not. 40 is from early 2010's, the 202/402 is from 2017. Also the Attiny20, which is less pins than the 40. But covered by my programmer. I would lean towards the 10 or the 20. 40 is too much IO.

Attiny10 has I2C on board (TWI). I'm going to stick with the 10. Keep it simple.

1.4.5 Present Questions

Questions: is a two pin UART the best I can expect to find for simple rf comms is lora a reasonable solution here? I want as low power as possible, Lora is not the lowest but if I transmit rarely, and for a short time perhaps it won't matter.

Is there some other way to transmit data? I don't want to use ultrasonic waves, and IR seems to be too much current, as well as requiring line of sight / lens.

1.4.6 Plan of attack

I'm going to prototype with the Microchip RF modules for now, and see if I can get this working off a solar panel. I have an ADC, and with RESET disabled another IO. Goal is now:

Configure RF modules w/arduino.

Configure RF module (one Uno receiver, other tiny transmitter)

Attach any sensor that uses ADC. (temp sensor, resistor light or thermocouple, current sensor (monitoring power supply), and I have the ones I already tested today (but will hold off on - the accelerometer, and hall effect).

Desired applications - Temp of hot surface (boiler) monitoring - Solar current input monitoring.

As I've already worked with a current sense before in my battery project (Electronics_Projects_2019), I will use again the INA169. It seems to be low enough power for my needs, though I will double check in practice. The data sheet lacks obvious power dissipation figures, while sensing, though quiescient is about 50uA.

1.4.7 Other Sensors

I need to verify that the following sensors could be used with low power:

Light dependent Resistor / Diode

Sound sensor (mic)

1.4.8 RF Searching

RF Transceiver ICs require assembly/programming/time, so we want RF Transceiver modules instead. There are a number of roughly 10mA TX active, but that is the lowest I can find. Among these ics (not modules), most are SPI, have their own ARM core... The modules are a bit better in having some that can be controlled by serial. I like the TRM-433-LT, but I will skip this for now ($20 each).

I also like the RC11xx-RC232, though the latter is about 30mA TX. That may be the best I can hope for. These two are 433MHz (ISM is 433 - 434MHz). The radiocrafts product is essentially a

simplified solution, preprogrammed with a uart which can be used to adjust settings. This option appears to be on part with the microchip offering I've looked at, the RN2483, in fact they both came up on my search results, next to each other. RN doesn't have what IC they use but RC, uses CC1110.

Another option:

ZETAPLUS-433-SO is faster than others (500Kbs), and has lower TX (18mA). Up to 2KM range...

There's more options (over $18), but for the following specifications, the above three are basically what you can choose from. Ignoring those with high TX power rates (sparkfun), or requires SPI (stmicro) Specs:

- In stock
- Frequency 433-434MHz
- battery powered vcc range (2)

[9]

I will start with two Radiocraft modules. Let's see how that plays out.[10]

[9] There's only 25.

[10] A set of Dev boards for the radiocraft are $250. Ouch.

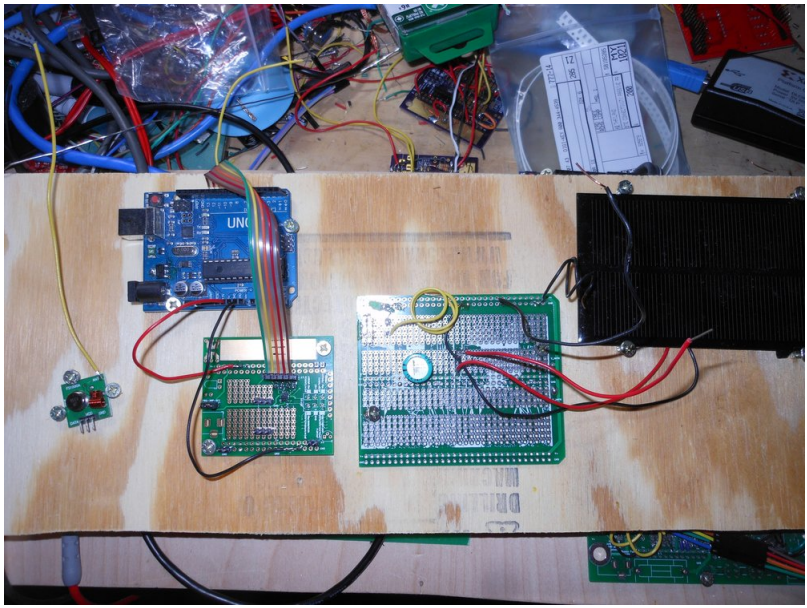

Figure 3: RF Transmitter w/ PCB Rev 4

1.4.9 PCB Rev 5

- Added instructions to back of board to speed up getting back to this project after some time.
- Moved resistors to be further away from programming header
- Fixed missed connection from JMP to V1 / V2 when I moved barrel plug, and added back the bottom right mounting hole in rev4.
- Changed prg/jmp/vcc to v1/jmp/v2 (either v rail can be used for programming)

1.4.10 Input Protection on Accidental Reverse Cable Hookup

I'm going to want something to block plugging in the dupont cable backwards from breaking things. I had a square wave outputting on

one pin of the Attiny10, and programming was going well, with the cable left in, until I removed, then connected it in backwards. The square wave blew something on the Atmega328p of the uno. This led me to quick replacement of the Uno atmega's here: electronics_projects_2020/AVRdragon_Optiboot_Atmega328
I don't know what's better. Make it foolproof, or let people learn the hard way. With the latter, they at least get a way to reprogram blank Atmegas (quite useful with Unos).
Perhaps instead, a mated connector would be enough on the breakout...

1.4.11 Nuts & Volts 433MHz RF Transmitters

A recent nuts and volts issue (2019), covered a few sections on RF in one magazine. They covered, transmitters, as well as measuring power output of a transmitter (in separate articles). The RF transmitters they covered were 433MHz, ebay transmitters. The transmitters output the square wave you feed to them. Extremely basic. But also open to modifications and adaptations. The part count is very low, with minimal ICs. (The receivers do have an IC, though.) I've bought some of these, and will do some testing. One benefit (or drawback) of these, is that they output whatever you feed them. They don't require UART.

Figure 4: simple rx tx on 433mhz

1.5 Connecting to Programmer Tips

When connecting, have a base setup that works as a sanity check. Then work on a second setup.

It may be possible to blow out atmega328p pins if you leave attiny running, then plug in the cable back to front (i.e. pin 13 in 9).

Make sure the cable is connected well. Loose cables, will cause the programmer to fail. Solder wires to Uno if necessary.

1.6 Attiny10 Timer Fail

Note: this is a segway, where I tried to get a 6KHz clock out of one of the Attinys for the 60Hz project).

The goal was to get a 6KHz square wave out of the Attiny10 (first using the timer). After a night of trying to wrangle with timer 0 and CTC mode of the Attiny10 timer (which failed to work as expected). I've decided to buy a 6KHz oscillator. I was able to get 60.3KHz out of the Attiny using no optimization in mplab and IO as fast as possible. I was able to get 6.13Khz using optimization and a for loop (unfortunately the nops got optimized out).
I learned that the cable I was using to program the attiny10 was giving me connection trouble. I soldered one side to an uno. That out of the way, When programming the attiny10 with my protoboard, it's as easy as, 1) connect cable to protoboard, 2) press P in serial monitor of Uno, 3) paste hex contents of mplab build into serial monitor. No need to ever reset the uno, even if it doesn't print that it detects the attiny. It's pretty good about this, and if connected correctly, will just write to the chip, when P is pressed. Then disconnect the cable, and the sketch will run. When it's time to reprogram, reconnect the cable, hit P, and paste contents, etc... **End Segway.**

1.6.1 AVR-objdump -S

Here's some code from lowpower_tx433mhz_2/3:

```
1                PORTB = bitRead(*strctPtr, i) << 2;
2    8c:   40 a1          lds   r20, 0x40
   ; 0x800040 <__DATA_REGION_ORIGIN__>
```

```
3    8e:    51 a1            lds      r21, 0x41
   ; 0x800041 <__DATA_REGION_ORIGIN__+0x1>
4    90:    62 a1            lds      r22, 0x42
   ; 0x800042 <__DATA_REGION_ORIGIN__+0x2>
5    92:    73 a1            lds      r23, 0x43
   ; 0x800043 <__DATA_REGION_ORIGIN__+0x3>
6    94:    08 2f            mov      r16, r24
7    96:    04 c0            rjmp     .+8
   ; 0xa0 <main+0x38>
8    98:    76 95            lsr      r23
9    9a:    67 95            ror      r22
10   9c:    57 95            ror      r21
11   9e:    47 95            ror      r20
12   a0:    0a 95            dec      r16
13   a2:    d2 f7            brpl     .-12
   ; 0x98 <main+0x30>
14   a4:    41 70            andi     r20, 0x01
   ; 1
15   a6:    44 0f            add      r20, r20
16   a8:    44 0f            add      r20, r20
17   aa:    42 b9            out      0x02, r20
   ; 2
18           #else
19                   //round up by default
20                   __ticks_dc = (uint32_t)(ceil(fabs(__tmp)))
21           #endif
22
23           __builtin_avr_delay_cycles(__ticks_dc);
24   ac:    43 e0            ldi      r20, 0x03
   ; 3
25   ae:    4a 95            dec      r20
26   b0:    f1 f7            brne     .-4
   ; 0xae <main+0x46>
27   b2:    00 00            nop
28   b4:    8f 5f            subi     r24, 0xFF
   ; 255
```

```
29    b6:    9f 4f           sbci    r25, 0xFF
    ; 255
30        uint8_t i = 0;
31
32        while(1)
33        {
34
35            for(i=0;i<32;i++){
36    b8:    80 32           cpi     r24, 0x20
    ; 32
37    ba:    91 07           cpc     r25, r17
38    bc:    39 f7           brne    .-50
    ; 0x8c <main+0x24>
39    be:    e4 cf           rjmp    .-56
    ; 0x88 <main+0x20>
```

It's easily readable assembler. See at the end, are the two jumps back in the loop 0x8c (for loop), and 0x88 (while 1). See the read on PORTB, and see the delay, which was _delay_us(10), but expands to the avr library definition.

1.7 Attiny10 w/Solar and w/433MHz TX

There was an issue with 433MHz_1 and 2 that the output was not on time. I have a 32 bit struct, and when reading the bits, the first 10 bits slowly ramp up from say 500us to 2ms for a 1 or 0. This was resolved by putting in a delay in between reads.

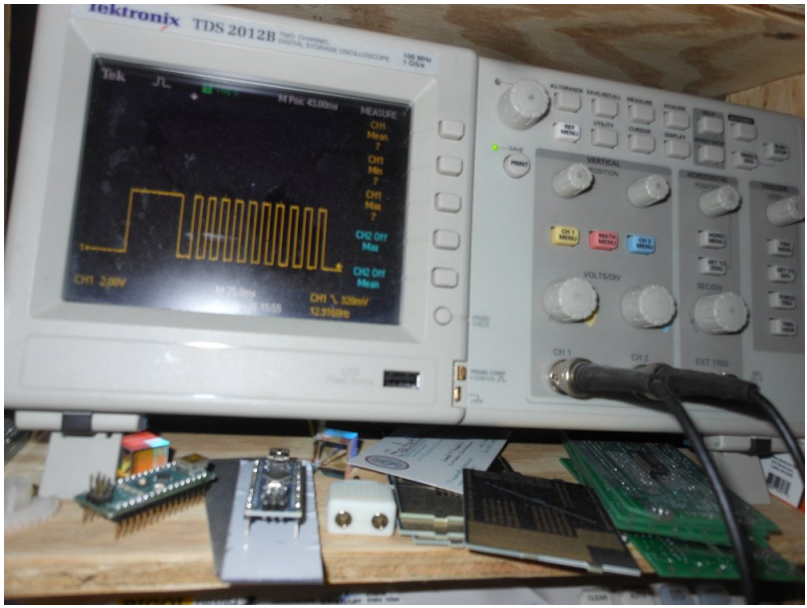

Figure 5: With a _delay_ms, there is no ramping up. But, the delay had to be a minimum, otherwise some ramping did occur. It's very slow, but usable for things without a lot of data to transmit.

Some code revision notes:
433MHz_3: 128000 KHz, 3ms delay in between bit reads being output to PORTB ¿¿ 2 (PB2). Timing is now perfect, albeit slow.
433MHz_4: 8Mhz: 60us delay in between bit reads. faster, but has a slight time difference. RX seems to not be working right.
I have 433MHz #4 running under a LED light w/solar panel. 1.4Volts on the Attiny and its able to transmit successfully (when the rx is right next to it on the bench). 1.3Volts and noise appears on RX. So about 1.4Volts is the lower limit... Perhaps.
433MHz_7: I confirmed in 6 and 7 that 8MHz is not working (at least consistently). Sticking with 128KHz for now. Seems to work as long as all wires are connected. Sometimes I power cycle if something is knocked.
433MHz_8: adding sleep in place of delay_ms(100). Will have Watchdog power back on.

1.7.1 433MHz Module TX Max Speed

2KHz was OK. Beyond that, the RX'd signal goes out of phase, i.e. can't keep up. 5KHz is unusable. Tested with sig gen.

1.8 Further Notes

https://www.eevblog.com/forum/microcontrollers/powering-devices-via-gpio-pins/msg2720044/#msg2720044 - Using GPIO to power devices.
leonerd TODO LINK HERE (bookmarks in main mach) attiny815 with rf. notice the coiled antennas
https://www.eevblog.com/forum/beginners/rf-very-low-power-comms-simple/msg3016400/#msg3016400 - Forum post regarding this project.
https://trmm.net/ATtiny10 - Programming Attiny10 with free / open source toolchain
https://blog.podkalicki.com/attiny13-blinky-with-timer-compa/ - Similar example for Attiny13, adapted but did not work with Attiny10.