Objective:

Get distance data from the MB1005.

*ParkSonar-EZ sensor features an easy to use logic level (high/low) output, and RS232 format serial output. *Factory calibration and testing is standard.*

Features

- Proximity vehicle detection
- Simultaneously runs along side other nearby sensors
- ~10 second object acquire time
- ~5 second object release time
- Range information available on Pin 5 to 254 inches → 6 inches - 254 in
- 2.5V to 5.5V supply with 2mA typical current draw
- Interfaces are simultaneously active
- Serial, 0 to Vcc, 9600 Baud, 81N 8N1?
- Digital logic High/Low (True/False) output
- Continuously variable gain for side lobe suppression
- Free run operation continually measures and outputs proximity information
- Sensor operates at 42KHz
- Actual operating temperature range from –40°C to +65°C, Recommended operating temperature range from 0°C to +60°C

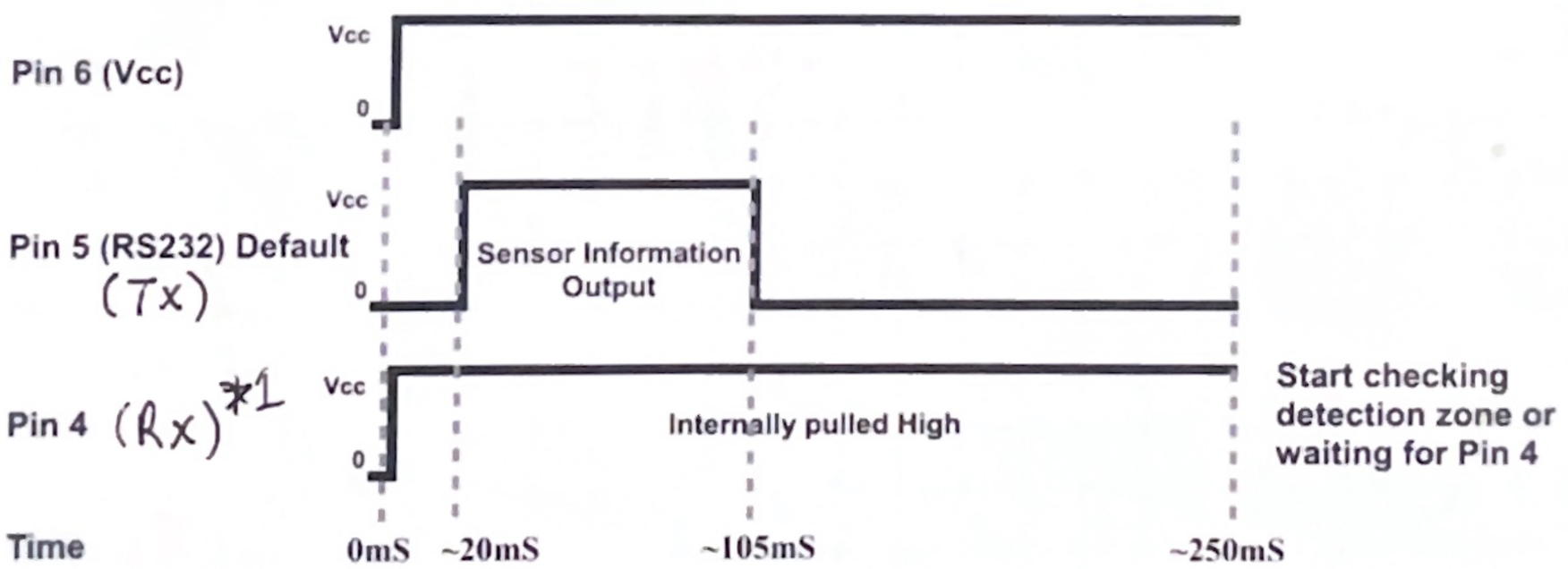
This sensor uses a UART, but RS232 instead of TTL.

RS232 sends the signal fliped compared to TTL.

The data sheet suggests a converter board but we will use a transistor instead.

*1 Notice that the Rx isn't sending commands, but instead just uses a high or low to activate the chip.

Power-Up Timing Diagram

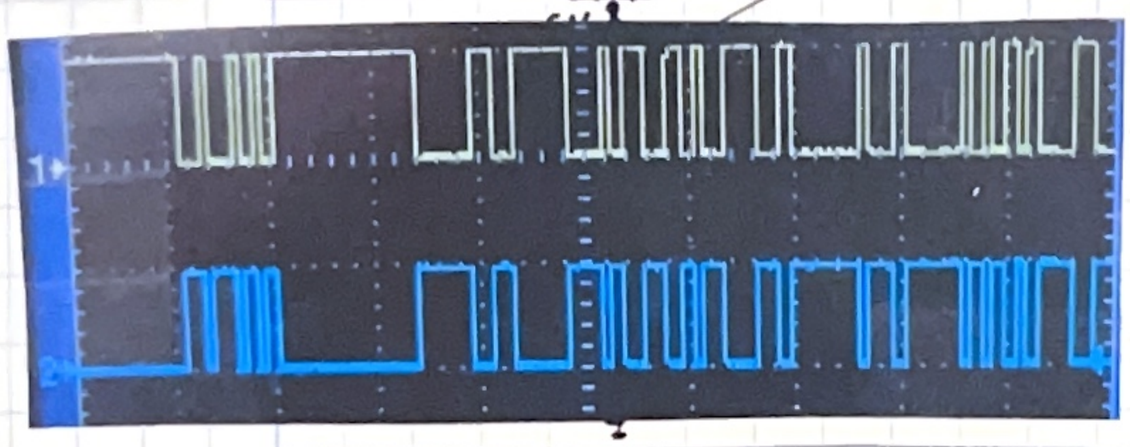


MB1005 ParkSonar Sensor Continued

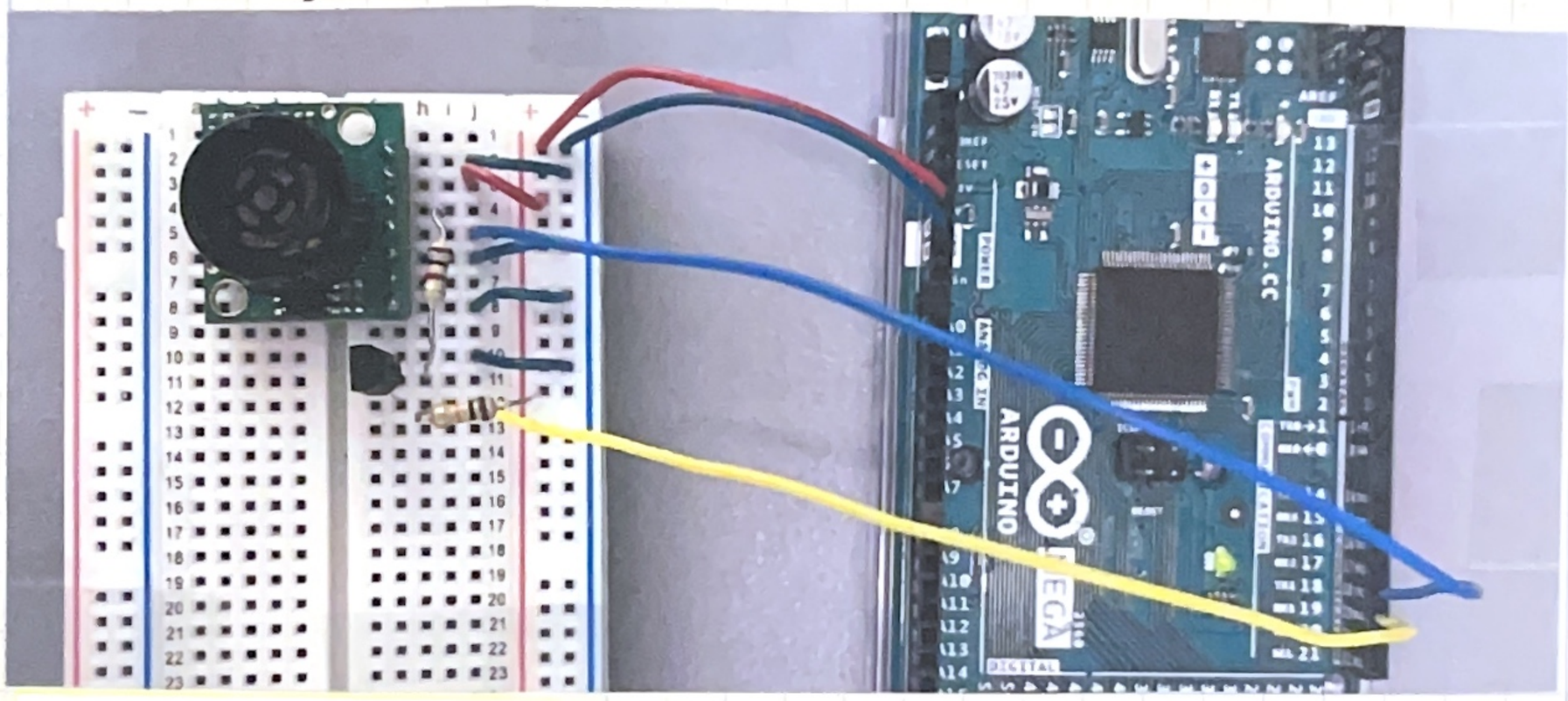
MB1005 Connections to Arduino Mega

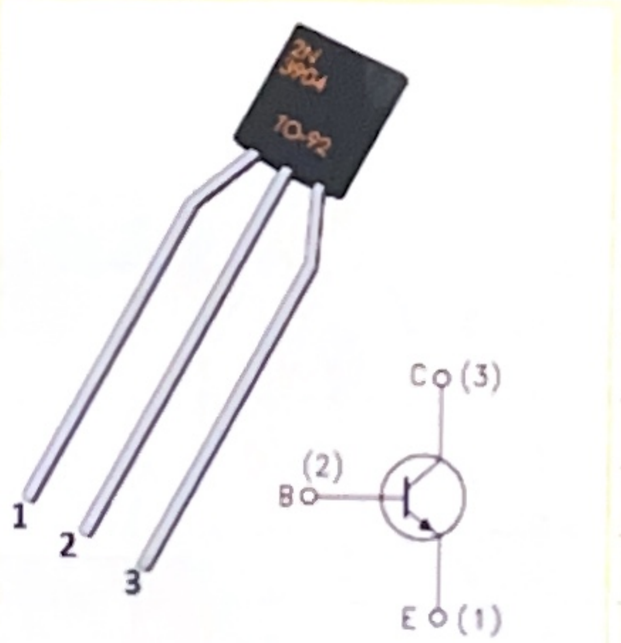
GND	GND
+5V	+5V
Tx	Inverter Circuit
Rx	18 (Tx1)
AN	GND
PW	NC
BW	GND

Inverter Circuit



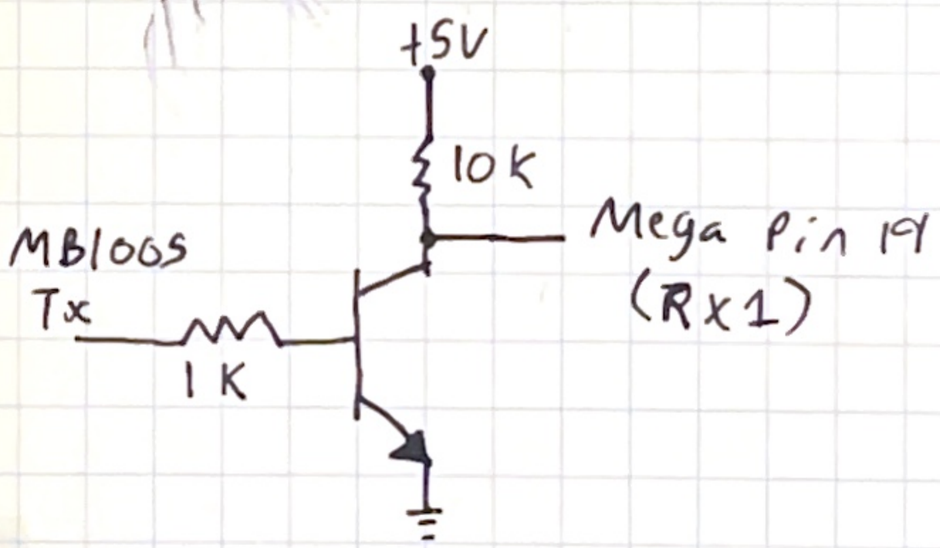
Channel 2 shows RS232.





2N3904	
1	Emitter
2	Base
3	Collector

Inverter Circuit



MB1005 ParkSonar Sensor Continued

Pin 5-TX - The TX output delivers asynchronous serial with an RS232 format, except voltages are 0-Vcc. If a target is detected at 8 inches the output appears as follows: "R008 P1<carriage return>". The output is an ASCII capital "R", followed by three ASCII character digits representing the range in inches up to a maximum of 255, followed by an ASCII space and the ASCII character "P", follow by one ASCII digit "1 or 0" corresponding to the proximity information, followed by a carriage return. Range information is provided for reference. Although the voltage of 0-Vcc is outside the RS232 standard, most RS232 devices have sufficient margin to read 0-Vcc serial data. If standard voltage level RS232 is desired, invert, and connect an RS232 converter such as a MAX232.

The output on the Tx of the MB1005 seems to always be 8 bytes. We plan to read the range number but only print it out if we also see the carriage return. This is kind of error checking it, but not really. It may help keep the serial monitor working properly.

```
char range_data[3];
int range;

void setup()
{
  // Turn on our Serial monitors.
  Serial.begin(115200);
  Serial1.begin(9600);
  pinMode(18, OUTPUT);
  // Take RX HIGH to enable the sensor.
  digitalWrite(18, HIGH);
}

void loop()
{
  // Wait for an entire packet to arrive
  while(Serial1.available()>8)
  {
    // If we see 'R' to indicate the range is coming
    if (Serial1.read() == 'R')
    {
      for( int i=0; i<4; i++)// Read Range data
      {
        range_data[i] = Serial1.read();
      }
    }
    Serial1.read(); // Read the space
    Serial1.read(); // Read the 'P'
    Serial1.read(); // Read the 0 or 1 (I'm ignoring these)

    if (Serial1.read() == 0x0D) // If we see the carriage return
    {
      range = atoi(range_data); //convert ASCII to integer
      Serial.print("The range is "); Serial.print(range); Serial.println(" inches.");
    }

    // Clear the buffer
    while(Serial1.available() > 0)
    {
      Serial1.read();
    }
  }
}
```

```
The range is 7 inches.
The range is 7 inches.
The range is 7 inches.
The range is 24 inches.
The range is 24 inches.
The range is 24 inches.
```

Weird things happen without this. ~~What~~ Watch Serial.available() climb up if you don't use it.