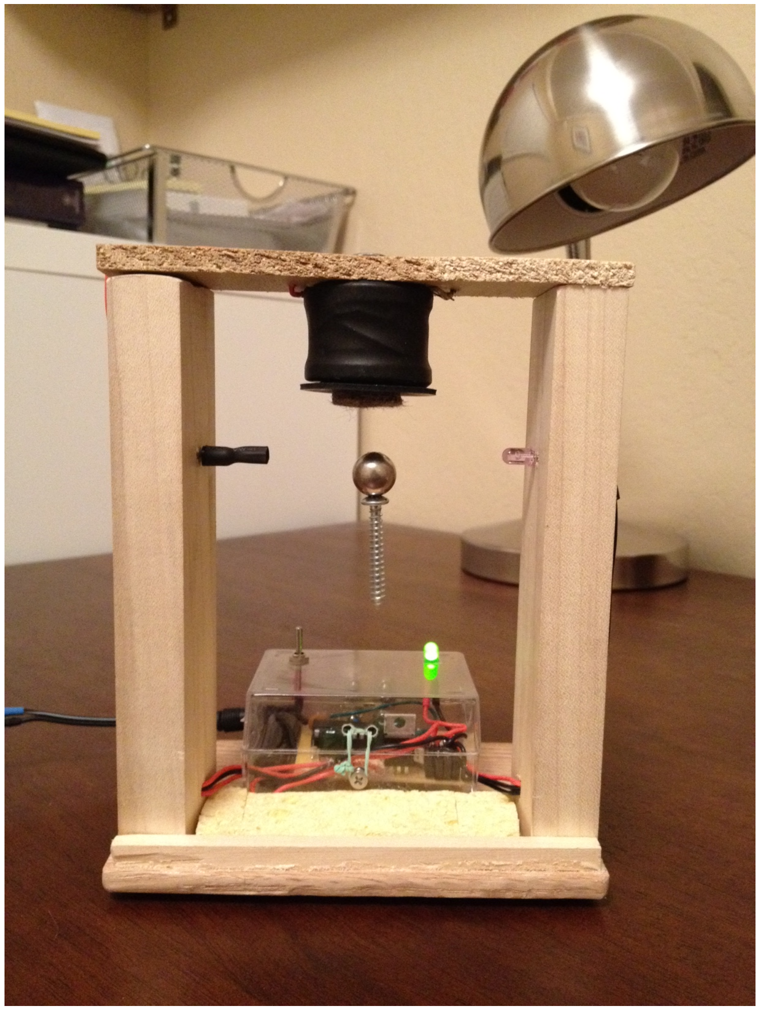
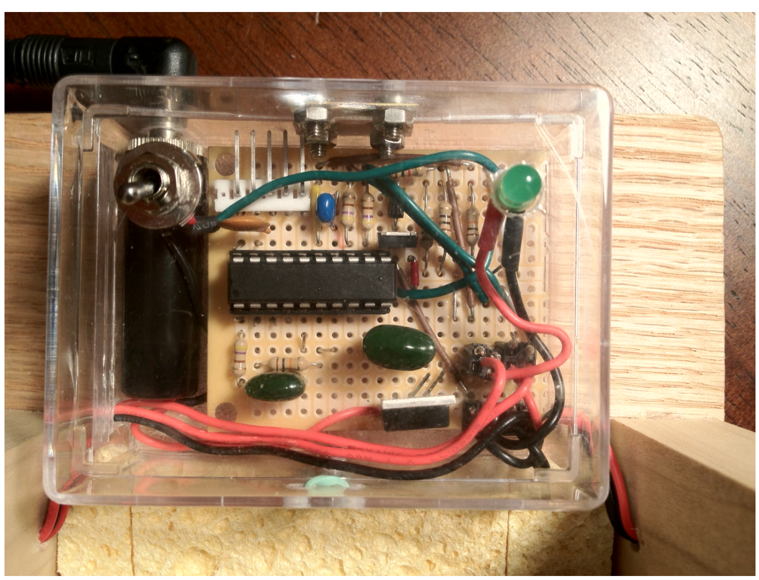
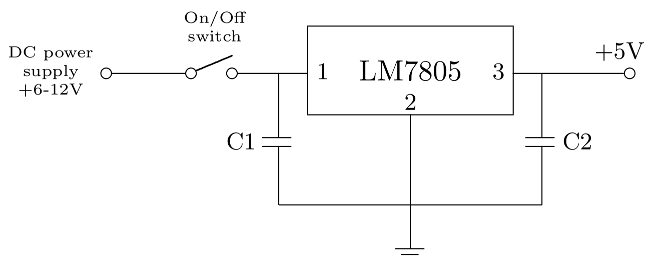
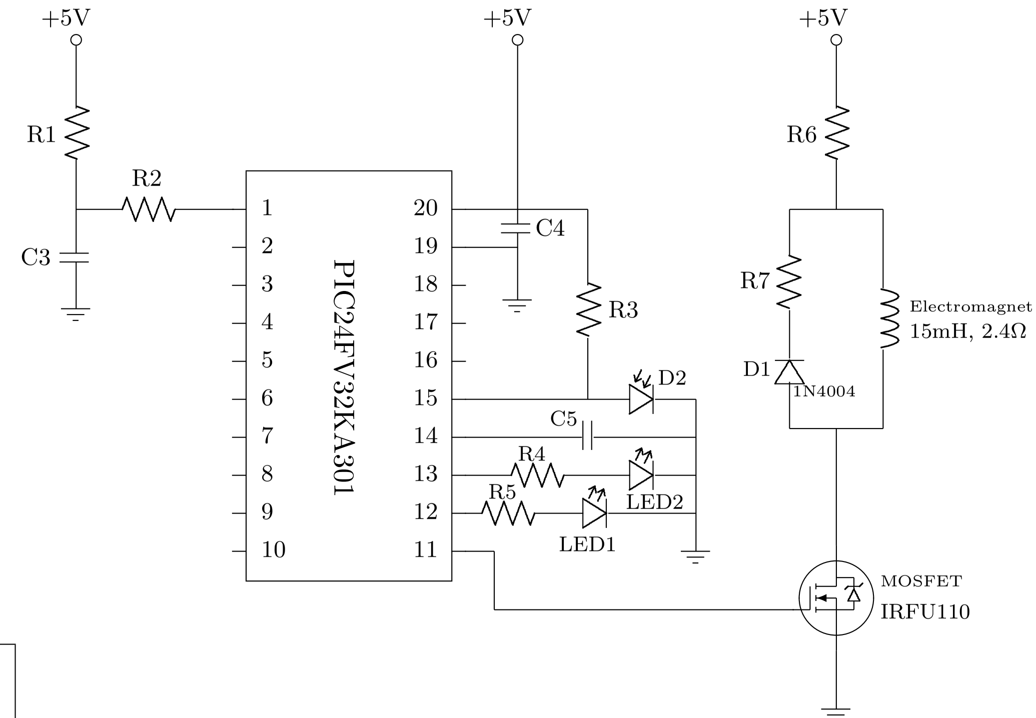
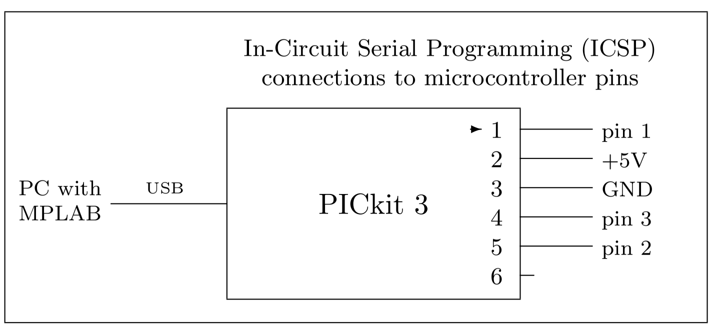




Circuit schematic







R1	10KΩ	C1	0.22μF
R2	470Ω	C2	0.1μF
R3	10KΩ	C3	0.1μF (ceramic)
R4	470Ω	C4	0.1μF (ceramic)
R5	470Ω	C5	10μF (ceramic or tantalum)
R6	5Ω	D2	Infrared photodiode
R7	1KΩ	LED1	Infrared emitter
		LED2	Status indicator

Microcontroller source code

```
/*
** file: levitator.c
** target: PIC24FV32KA301
** compiler: C30 version 3.30
** IDE: MPLAB X
*/
#include <p24Fxxxx.h>

_FBS(BSS_OFF & BWRP_OFF)
_FGS(GWRP_OFF & GSS0_OFF)
_FOSCSEL(IESO_OFF & LPRCSEL_HP & SOSCSRC_ANA & FNOSC_FRC)
_FOSC(FCKSM_CSDCMD & SOSCSEL_SOSCLP & POSCFREQ_MS & OSCIOFNC_OFF & POSCMOD_NONE)
_FWDT(WINDIS_OFF & FWDTEN_OFF & FWPSA_PR128 & WDTPS_PS32768)
_FPOR(MCLRE_ON & BORV_V20 & I2C1SEL_PRI & PWRTEN_OFF & LVRCFG_OFF & BOREN_BOR0)
_FICD(ICS_PGx3)
_FDS(DSWDTEN_OFF & DSBOREN_OFF & DSWDTOSC_LPRC & DSWDTPS_DSWDTPSF)

void delay(unsigned int n) {
    unsigned int j;
    for (j=0; j<n; j++);
}
unsigned int read_adc() {
    AD1CON1bits.SAMP=1;             // Start the ADC Sampling
    while (AD1CON1bits.DONE != 1); // Wait for ADC conversion
    return ADC1BUF0;               // Return the 12-bit ADC Result
}

int main ()
{
  // Initial the Ports Used
  TRISB  = 0b0001000000000000;  // Set RB12 and RB13 to Input and all other RBx bits to Output
  ANSBbits.ANSB12 = 1;          // Set AN12/RB12 as Analog Input
  // Configure analog-to-digital converter
  AD1CON1bits.ADSIDL = 0;       // Discontinue module operation when device enters idle mode
  AD1CON1bits.FORM = 0b00;      // Data output format: absolute decimal result, usigned, right-justified
  AD1CON1bits.SSRC = 0b0111;    // Conversion starts automatically after sampling finished
  AD1CON1bits.ASAM = 0;         // Sampling begins when SAMP bit is manually set
  AD1CON1 |= 0x0400;            // Enable 12-bit ADC conversion (default is 10 bits)
  AD1CON2bits.PVCFG = 0b00;     // Positive voltage reference: AVDD
  AD1CON2bits.NVCFG = 0b00;     // Negative voltage reference: AVSS
  AD1CON2bits.OFFCAL = 0;       // Inverting and non-inverting inputs of channel S/H are connected to normal inputs
  AD1CON2bits.BUFREGEN = 0;     // A/D result buffer is treated as a FIFO
  AD1CON2bits.CSCNA = 0;        // Do not scan input selections for CH0+ during SAMPLE A bit
  AD1CON3bits.ADRC = 1;         // AD conversion clock source bit: RC clock
  AD1CHSbits.CH0NA = 0b000;     // Sample A channel 0 negative input select bits: AVSS
  AD1CHSbits.CH0SA = 0b01100;   // S/H amplifier positive input select for MUX A multiplexer setting bits: AN12
  AD1CON1bits.ADON=1;           // Turn ON ADC Peripheral

  unsigned int adc_input, prev_adc_input, min_ir, max_ir, middle_ir, corrected_input;
  signed int highpass;

  // Read the IR sensor with the IR LED turned on and not turned on
  PORTBbits.RB8 = 0;
  delay(2000);
  min_ir = read_adc();
  PORTBbits.RB8 = 1;
  delay(2000);
  max_ir = read_adc();
  // We'll try to stabilize the system around the average of these two values
  middle_ir = min_ir/2 + max_ir/2;

  // Blink indicator LED twice (just for fun)
  PORTBbits.RB9 = 1;
  delay(60000);
  PORTBbits.RB9 = 0;
  delay(60000);
```

```
  PORTBbits.RB9 = 1;
  delay(60000);
  PORTBbits.RB9 = 0;
  delay(60000);

  // The main program loop
  for(;;) {
    prev_adc_input = adc_input;
    adc_input = read_adc(); // Read the sensor

    // We use a high pass filter to correct the sensor reading, taking the magnet's velocity into account.
    // This is necessary to achieve stable levitation!
    highpass = (highpass+(signed int)(adc_input-prev_adc_input))*3/4;
    corrected_input = adc_input + 48*highpass;

    // The indicator LED displays the uncorrected reading (makes for a nicer display)
    if (adc_input <= middle_ir)
        PORTBbits.RB9 = 1;
    else
        PORTBbits.RB9 = 0;

    // This pin controls the electromagnet
    if (corrected_input <= middle_ir)
        PORTBbits.RB7 = 1;
    else
        PORTBbits.RB7 = 0;
  }
  return 0;
}
```

Notes

1. The recommended value for the capacitor C1 according to the LM7805 application notes is 0.33μF. I used a smaller 0.22μF capacitor since that's what I had at the moment, and it works.

2. The capacitors C2 and C4 both do the same thing, so I assumed one of them was unnecessary and only used C4 in the final circuit.

3. I used two 10Ω resistors in parallel in place of the 5Ω resistor R6.

4. Programming the microcontroller using PICkit 3 was a bit unreliable. I was only able to program the microcontroller by selecting the option to power the circuit from USB in MPLAB X.

5. The current consumption of the system is around 0.5A (= 3W of power with a 6V power supply) when the electromagnet is turned fully on, i.e., when no magnet is detected between the IR LED emitter and sensor, and 0.3A (= 1.8W of power with a 6V power supply) during levitation. The frequency of the switching signal was measured using a multimeter to be around 2KHz.

6. The screw or other vertical object attached to the levitating magnet increases stability. It seems harder to stabilize a stand-alone magnet (probably because of its much lower moment of inertia).