Soirée Pratique

Bike remote project
19/10/2015

slides + extra information:
www.ieee-sb-leuven.be/soireepratique2015
Facebook group:
Soirée Pratique Leuven

IEEE Student Branch Leuven

Today's session: bike remote

- play sounds/lights on your bike using an IR remote

- today:
  - provide schematics box cover
  - adding IR remote to existing circuit
  - program sounds in arduino corresponding to button presses

Schedule sumo robot competition

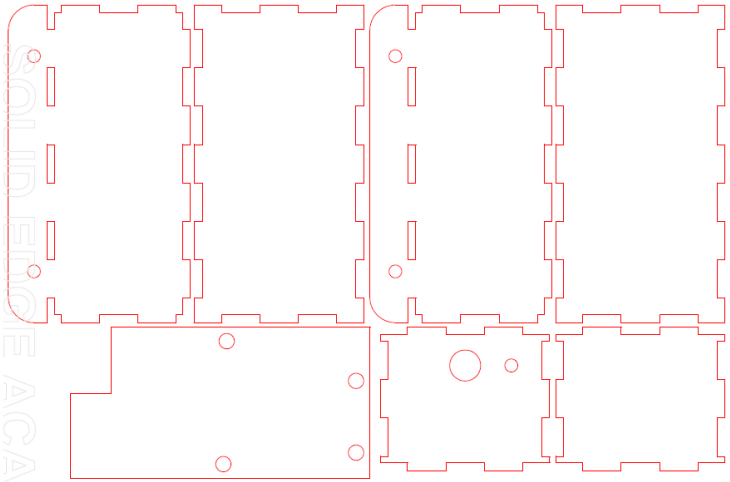
- Tuesday 3/11 at 19h30:
  - first session sumo robots: introduction arduino, information on ordering parts
- Monday 16/11:
  - motor session: the “muscles”
- Monday 30/11:
  - sensor session: the “eyes”
- Second semester:
  - frame building: the “skeleton”
  - advanced programming: the “brains”
  - training sessions + final competition

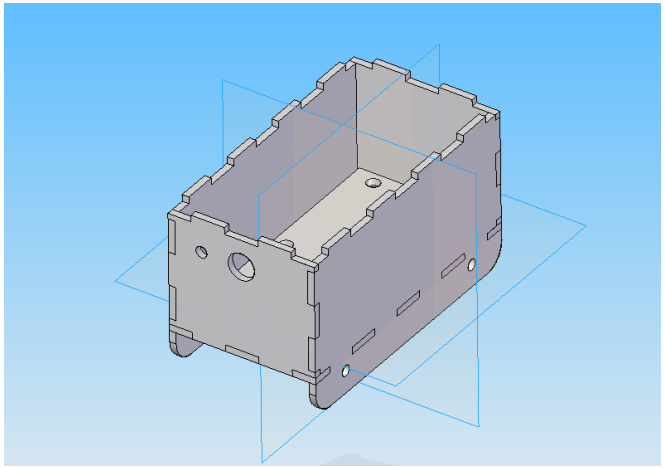
Bike remote components

- components last time:
  - arduino uno
  - leds /resistors
  - breadboard / cables
- new components:
  - **IR transmitter/receiver**
  - **buzzer**
  - **battery connector**
- components can also be used for sumo robot competition
  non-IEEE members: 15 euro
- IEEE members: **only 2 euro's!**

Box cover to mount on bike

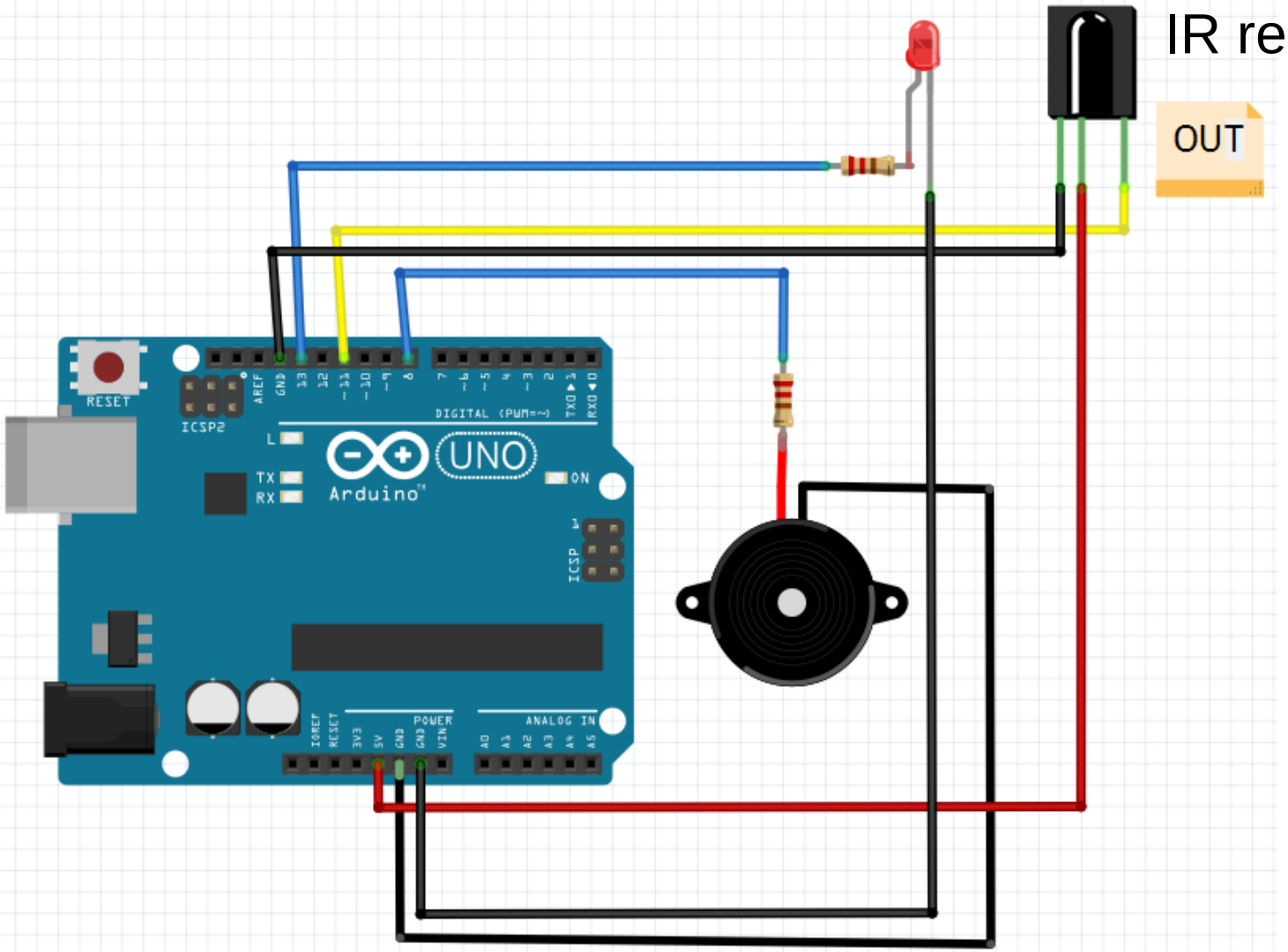
- files available at: www.ieee-sb-leuven.be/soireepratique2015
- laser cutter at FabLab (mech departm)
- material used: plexiglas 3 mm





- glue (at FabLab) or tape sides together

Circuit schematic bike remote



IR receiver

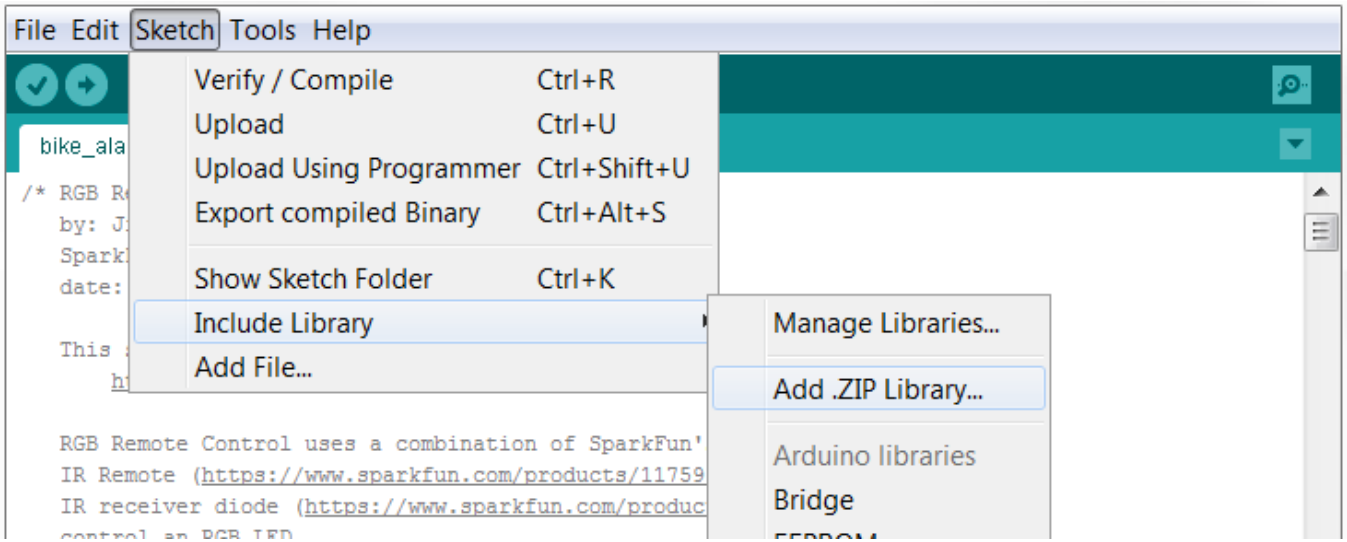
solder 220Ω resistor to led

220Ω resistor for buzzer is optional

IEEE Student Branch Leuven

Arduino library for IR remote

- "bike alarm basic" code available at www.ieee-sb-leuven.be/soireepratique2015
- add library "IRremote" in Arduino:



- navigate to folder "IRremote", then press "Open"

IR remote code: basic

- start from downloaded “bike alarm basic” code

- program yourself:
  1. turn led on with one button, off with another
     hint: use variable “ledState”
  2. play different tones with different buttons
     hint: use variables “playnote” and “this_note”
  3. play the defined “melody” using a button

- problem:
  - melody can’t be interrupted since we’re using delays

IR remote code: advanced

- add **interrupts** to code in order to stop melody during play
  - also useful for sumo robots to interrupt current behaviour
- this uses timer of the microcontroller

- Important F_CPU = 16MHz

IR remote code: interrupts

- Pause current actions and respond to “external” event
- Used for critical actions that need to be performed immediatly
- There is a hierarchy in interrupts
- We will look at timers only
- Powerful tool, if used carefully!!

IR remote code: interrupt timer

- Arduino Uno has 3 timers: named 0, 1 and 2
- If you start programming interrupts on a timer, other function who use that timer will no longer work 2
- ATTENTION: delay(), millis, micros() uses timer0, IRreceiver uses timer1, tone uses timer 2

IR remote code: timers

- Timer 0 and 2 have 8bit resolution (can count till 255 (zero included))
- Timer 1 is a 16bit resolution timer: can count untill 65535
- 2 timer interrupts possible:
  - Overflow: when counter overflows (255 for timer 0 and 2)
  - Compare: compare against a value (most be smaller than the overflow)

IR remote code: timer setup

- X indicates timer number: 0, 1 or 2
- TCCRxA / TCCRxB: Timer/Counter Control Register: set type of timer, how fast it works …
- TCNTx Timer/Counter Register: the current timer value
- OCRxA/B: Output Compare Register A/B
- TIMSKx: Timer/Counter Interrupt Mask Register to enable which interrupts are used

IR remote code: Timer/Counter Control Register

TCCRxA								
Bit	7	6	5	4	3	2	1	0
0x80	COMxA1	COMxA0	COMxB1	COMxB0	-	-	WGMx1	WGMx0
ReadWrite	RW	RW	RW	RW	R	R	RW	RW

TCCRxB								
Bit	7	6	5	4	3	2	1	0
0x80	ICNCx	ICESx	-	WGMx3	WGMx2	CSx2	CSx1	CSx0
ReadWrite	RW	RW	RW	RW	R	R	RW	RW

IR remote code: Timer/Counter Control Register

- Clock selectbit:

CSx2	CSX1	CSX0	Description
0	0	0	no clock
0	0	1	clock/1
0	1	0	clock/8
0	1	1	clock/64
1	0	0	clock/256
1	0	1	clock/1024
1	1	0	ext clock falling
1	1	1	ext clock rising

IR remote code: Timer/Counter Control Register (example)

- We wish a 1s timer with timer1 (16bit)
  - CPU frequency = 16MHz
  - Take 1024 as prescaler: 1 0 1
  - To increment counter takes 1/16e6*1024 s = 64 microseconds
  - 1second / 64 microseconds = 15625
  - 16 bit = 65535 so counter does not overflow (ok)

IR remote code: Timer/Counter and Output Compare Register

- TCNTx contains the current timer value,
- Can also be preset if needed (timer starts from preset value

- OCRxA contains the value of the timer/counter we wish to compare too

IR remote code: Timer/Counter Interrupt Mask

- TIMSKx: set what kind of interrupt should be active:
  - TOIEx: if this bit is set: overflow interrupt is active
  - OCIExA: if this bit is set: interrupt is send when OCRxA is equal to the timer register TCNTx

IR remote code: Timer/Counter Interrupt Mask

- TIMSKx: set what kind of interrupt should be active:
  - TOIEx: if this bit is set: overflow interrupt is active
  - OCIExA: if this bit is set: interrupt is send when OCRxA is equal to the timer register TCNTx

IR remote code: now implement the interrupt handle

- Use following function
- TOIEx: ISR(TIMERx_OVF_vect){
  }
- OCIExA: ISR(TIMERx_COMPA_vect){
  }
- Any variable inside this ISR functions need to be declared ‘volatile’ at beginning of file

IR remote code: how to set the right bits to timer registers?

- Use bitwise operators: |= (bitwise OR); &= (bitwise AND); ~ (NOT)
- Set timer to prescaler 1024: TCCR0B |= (1 << CS00 | 1 << CS02);
- Set timer to Clear Timer on Compare (reset when compared): TCCR0B |= (1<<WGM02);
- Stop timer: TCCR0B &= ~((1 << CS00) | (1 << CS01) | (1 << CS02))

IR remote code: interrupt code

```
// these variables will be used in program
volatile uint16_t counter = 1;
volatile uint16_t thisNote = 0;
volatile uint16_t counter_des = 0;

void setup() {
  cli(); // Disable all interrupts
  TCCR0A = 0;
  TCCR0B = 0;
  OCR0A = 156;
  TCCR0B |= (1<<WGM02);
  TIMSK0 |= (1 << OCIE0A);
  sei(); // Enable all interrupts
}

// Put this code where you want to start the melody
tuneSize = sizeof(melody2) / sizeof(int);
TCCR0B |= (1 << CS00);
TCCR0B |= (1 << CS02);

// Put this code where you want to stop the melody
TCCR0B  &= ~((1 << CS00) | (1 << CS01) | (1 << CS02))
```

IR remote code: interrupt code

```
ISR(TIMER0_COMPA_vect){ // Timer Interrupt Call (this code is executed when the interrupt is triggered)
  cli(); // disable all interrupts (we don't want other interrupts to interfere)

  if(counter >= counter_des){
    counter = 0;
    int noteDuration = (int)((1000 * (60 * 4 / tempo)) / durations2[thisNote] ); //duration in ms
    tone(buzzerPin, melody2[thisNote],noteDuration); // play tone(pin, note, duration)
    counter_des = (uint16_t) (noteDuration * 0.090); // next desired counter: the duration of the next note
is counted with timer0 this can count untill 156*1024/16*1 microseconds
    thisNote++; // increase note (next note of melody will be played)
    digitalWrite(LEDPIN,!digitalRead(LEDPIN)); // change led status (led will change according to melody

    if(thisNote > tuneSize){
      thisNote = 0; //reset note when melody is finished
    }
  }
  else{
    counter++; // increase counter
  }
  sei(); // enable all interrupts again
 }
```

Next Soiree Pratique

- Tuesday 3/11 at 19h30:
  - first session sumo robots: introduction arduino, information on ordering parts