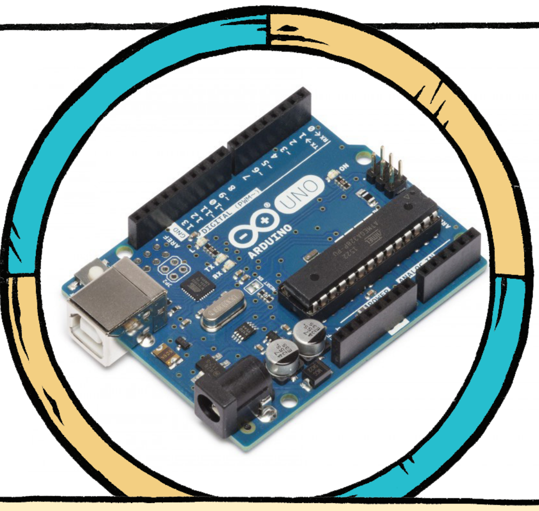
TOOL

ASSOCIATED SHEETS

analog inpouts



FAST MEASUREMENTS

How fast can we measure?

Sampling rate : 1 kS/s ; 9 kS/s ; 55 kS/s
Ease of use: medium to complicated
Use: high sampling-rate measurements

CONTINUOUS METHOD

Measurements, are constantly made and quickly transfered to the computer. The simplest method is to use the USB cable for this transfer, but communication between the board and the computer takes time. The serial monitor can work at different communication speeds: fast communication speeds are needed to obtain higher sampling rate.

```
void setup() {
  Serial.begin(115200) ;
}

void loop() {
  Serial.print(millis()) ;   // time is measured and transfered
  Serial.print("\t") ;      // tabulation character to separate the data
  Serial.println(analogRead(A0)) ;  // the voltage on A0 port is
}                                   // measured and transfered
```

The Arduino board must be configured to use the same communication speed as the serial monitor.

This program can run indefinitely, and sends the results continuously on the serial port. The sampling rate is of the order of 1.2 kS/s whit a communication speed of 115200 bauds. With the standard 9600 baud communication speed, the sampling rate drops to 90 S/s.

BURST METHOD

When the measurements are immediately transmitted to the computer, the transfer time slows down the process. This transfer time can be saved by first taking the measurements and then transferring them in packets. The sampling rate is increased, but the continuity of the measurements is lost. The small size of the memory of Arduino Uno board becomes limiting.

The program below works around 9 kHz. The limit therefore becomes the duration of the measurement burst. Indeed, the memory of the arduino Uno is limited, and beyond 800 integers, there is a risk of overflowing. This program measures 400 samples at a sampling rate around 9 kHz, for a duration of about 45 ms, transfers the data, and repeats itself.

```
const int NumberOfMeasures = 400 ;
int MeasuredVoltage[NumberOfMeasures] ;  // measurements array
int Time[NumberOfMeasures] ;       //  time array
void setup() {
  Serial.begin(9600) ;            // the communication speed can be standard
}

void loop() {
  for(int i = 0 ; i < NumberOfMeasures ; i++ ){ // measurements
    MeasuredVoltage[i] = analogRead(A0) ;
    Time[i] = millis() ;
  }
  for(int i = 0 ; i < NumberOfMeasures ; i++ ){ // data transfer
    Serial.print(Time[i]) ;
    Serial.print("\t") ;
    Serial.println(MeasuredVoltage[i]) ;
  }
}
```

- Some Arduino boards have larger memory;
- The space occupied by the time array can be saved by recording only the initial time and the final time.

HACKING THE ADC

With the previous method, the limit is the working frequency of the analog-to-digital converter of the card. This frequency can be increased at the cost of a slight deterioration of the quality of the measurement. The following link describes the method very well: https://sites.google.com/site/measuringstuff/the-arduino
The following program executes 800 measurements at 55 kS/s.

```
#define FASTADC 1
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= ~_BV(bit))
#endif
const int NumberOfMeasures = 800 ;
int MeasuredVoltage[NumberOfMeasures] ;  // measurements array
void setup() {
  Serial.begin(9600) ;            // the communication speed can be standard
#if FASTADC
  sbi(ADCSRA,ADPS2) ;
  cbi(ADCSRA,ADPS1) ;
  cbi(ADCSRA,ADPS0) ;
#endif
}
void loop() {
  unsigned long InitialTime = micros() ;
  for(int i = 0 ; i < NumberOfMeasures ; i++ ){ // measurements
    MeasuredVoltage[i] = analogRead(A0) ;
  }
  unsigned long FinalTime = micros() ;
  for(int i = 0 ; i < NumberOfMeasures ; i++ ){ // data transfer
    Serial.println(MeasuredVoltage[i]) ;
  }
  Serial.println(FinalTime - InitialTime) ; // duration of the measure
}
```