



# Introduction to Microcontrollers

Shivendu Bhushan

Sonu Agarwal

# Things to be covered today...

- Embedded System – Introduction, Examples
- Microcontrollers - basic features
- Input and output from a micro-controller
- Programming a micro-controller
- Interfacing Character LCD with Micro-controller
- How to use Infrared – Tsop sensor ?



# Embedded Systems

- Gadgets and devices
- Self controlled devices
- Contains I/O devices, storage devices and a central 'controller'

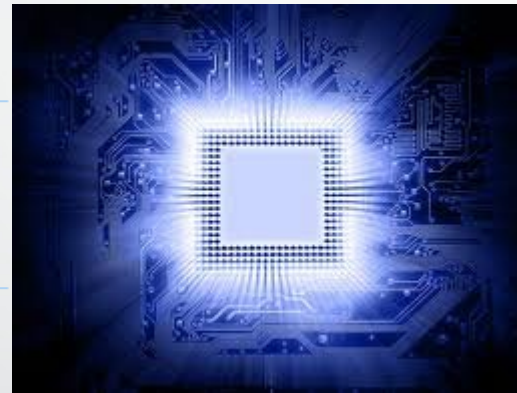
# Example: Music player



**Output**



**Input**



**Controller**

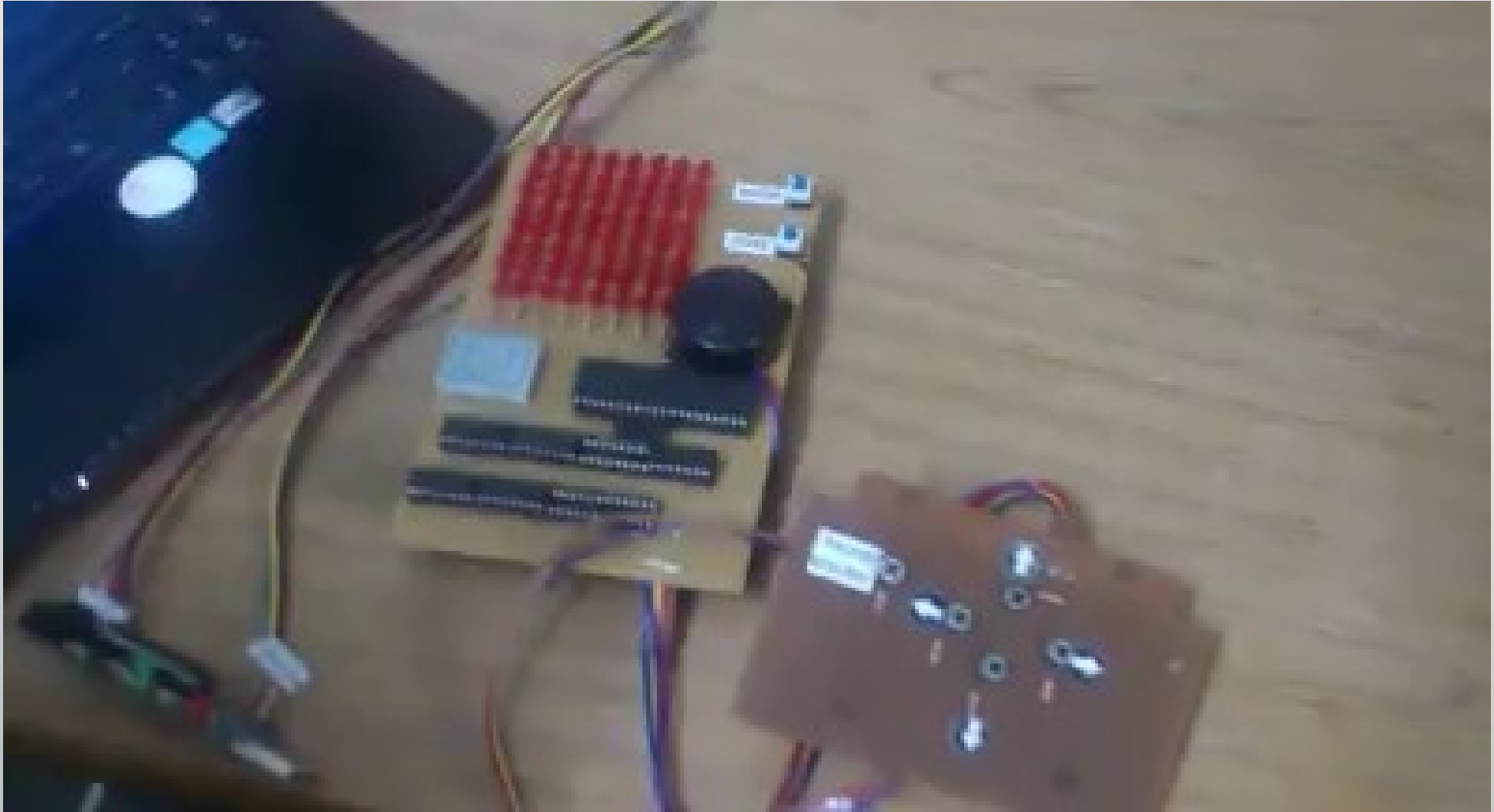


**Output**

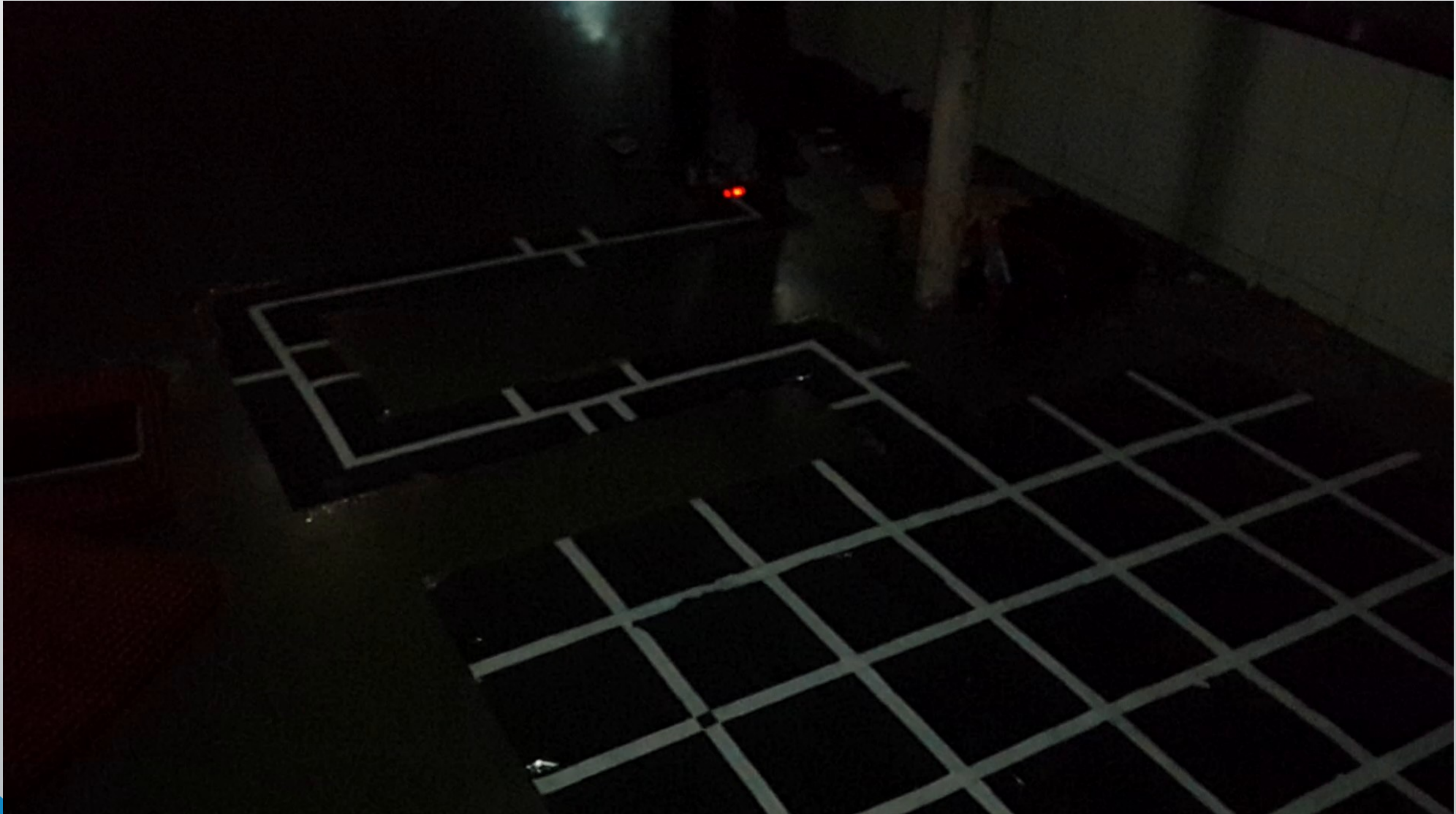


**Storage Device**

# Snake Game, Electromania 2013

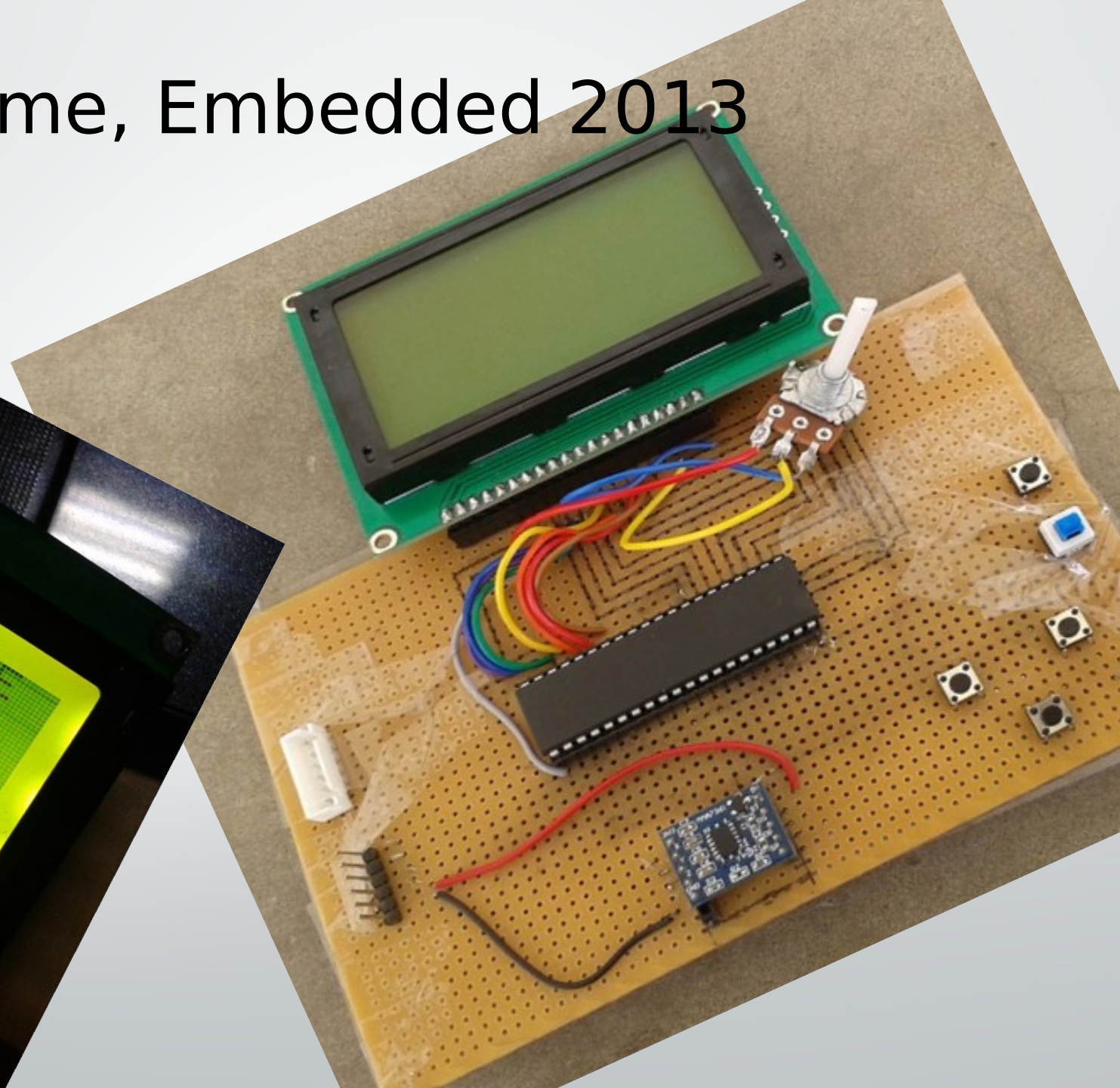


## Line Following Bot, Techfest 2014





# Maze Game, Embedded 2013





# Micro-Controllers

- Why “micro”? Larger controllers are available too: processors that run computers
- Out of several available vendors like Atmel, Intel, ARM, Cypress, etc. We will use Atmel ATmega microcontrollers
- Like computers they execute programs. We will use C as the coding language



# ATMEGA 8

- 28 pin IC
- 23 pins for I/O
- 5 pins reserved
- I/O pins divided into 3 groups of 8\* pins, called ports B, C and D
- Ports labelled as B, C and D

PDIP

(RESET) PC8	1	28	PC5 (ADC5/SCL)
(RXD) PD0	2	27	PC4 (ADC4/SDA)
(TXD) PD1	3	26	PC3 (ADC3)
(INT0) PD2	4	25	PC2 (ADC2)
(INT1) PD3	5	24	PC1 (ADC1)
(XCK/T0) PD4	6	23	PC0 (ADC0)
VCC	7	22	GND
GND	8	21	AREF
(XTAL1/TOSC1) PB6	9	20	AVCC
(XTAL2/TOSC2) PB7	10	19	PB5 (SCK)
(T1) PD5	11	18	PB4 (MISO)
(AIN0) PD6	12	17	PB3 (MOSI/OC2)
(AIN1) PD7	13	16	PB2 (SS/OC1B)
(ICP1) PB0	14	15	PB1 (OC1A)

# I/O Registers

- Input / Output is controlled through special variables called “**registers**”
- Registers are actual hardware memory locations inside the  $\mu$ Cs with predefined names and sizes
- Assigning a value to these registers in the program changes the corresponding hardware configuration. And, these values can be altered multiple number of time at any point in the program.
- There are 3 registers that control the I/O pins: **DDR, PORT and PIN.**
- Each port has it's own registers. Hence, **DDRC, PORTC, PINC** registers for port C; **DDRB, PORTB, PINB** for port B and likewise

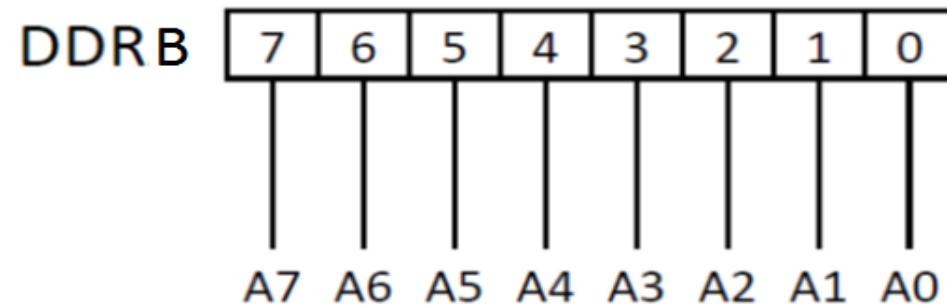
# DDR(Data Direction Register)

- Decides whether the pin is Input or Output
- DDR is an 8 bit register. Each bit corresponds to a particular pin on the associated port
- If a bit on the DDR register is 0, then the corresponding pin on the associated port is set as input
- Similarly, if the bit is 1, then the pin is set as output
- If a pin is configured as input, then it has some floating voltage unless an external voltage is applied
- For an output pin, the voltage is fixed to a particular value

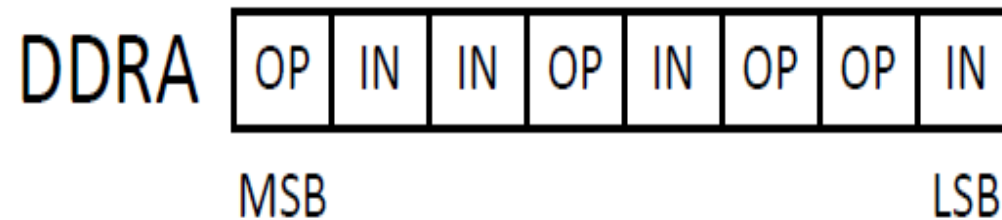


# Setting Register Values

- MSB of DDRB corresponds to the pin A7



- If DDRA = 0b10010110, then:





# PORT Register

- PORT is also an 8 bit register. The bits on the PORT register correspond to the pins of the associated port in the same manner as in the case of the DDR register.
- PORT is used to set the **output** value.
- If the pin is set as **output**, then a PORT value of 1 will set voltage at that pin to 5V, and PORT value 0 sets the voltage to 0V.
- If the pin is configured as an **input**, PORT value serves the purpose of **pull up** or **pull down**.

# PIN Register

- PIN is a register whose value can be read, but cannot be changed inside the program.
- It gives the value of the actual voltage at a particular pin. 1, if the value at the required pin is 5V and 0 for 0V.

# Summary

DDR = 0		DDR = 1	
PORT = 0	PORT = 1	PORT = 0	PORT = 1
Pin is input. If unconnected, <b>PIN</b> is 0.	Pin is input. If unconnected, <b>PIN</b> is 1.	Pin is output, value is 0. <b>PIN</b> is always equal to <b>PORT</b>	Pin is output, value is 5V. <b>PIN</b> is always equal to <b>PORT</b>

# Some C concepts

- `|` is bitwise OR. Eg. `10100111 | 11000101 = 11100111`
- `&` is bitwise AND. Eg. `10100111 & 11000101 = 10000101`
- `~` is bitwise NOT. Eg. `~10100110 = 01011001`
- `<<` is shift left. `>>` is shift right





# Simplest C program for a micro-controller

```
int main(){  
    return 0;  
}
```

# Example Program 1

```
#include <avr/io.h>
int main(){
  DDRA = 0b11111111; // or 255 or 0xFF
  while(1){
    PORTA = PINC;
  }
  return 0;
}
```

# Example Program 2

```
#include <avr/io.h>
#include <util/delay.h>
int main(){
    DDRA = 0xFF;
    while(1){
        PORTA = 0xAA;
        _delay_ms(1000);
        PORTA = 0x55;
        _delay_ms(1000);
    }
    return 0;
}
```

# How to Program MCU?



CVAVR/AVRSTUDIO

o-> **HOW?**



**AVRSTUDIO**

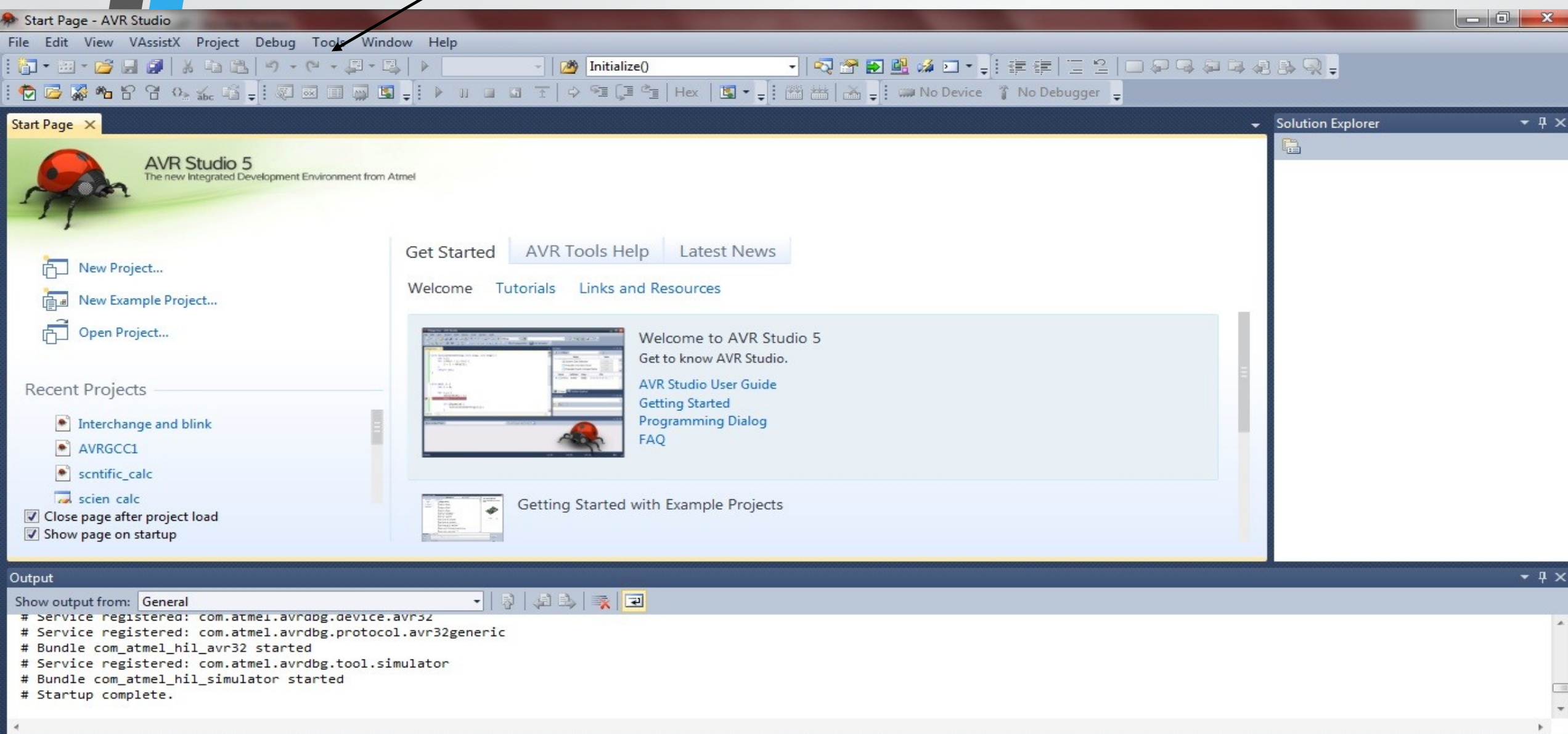
#Problem: What kind of files MCU can execute ?  
#Problem: How to transfer that file to MCU ?



# AVR Studio



# Select Tools



# Select Add STK500

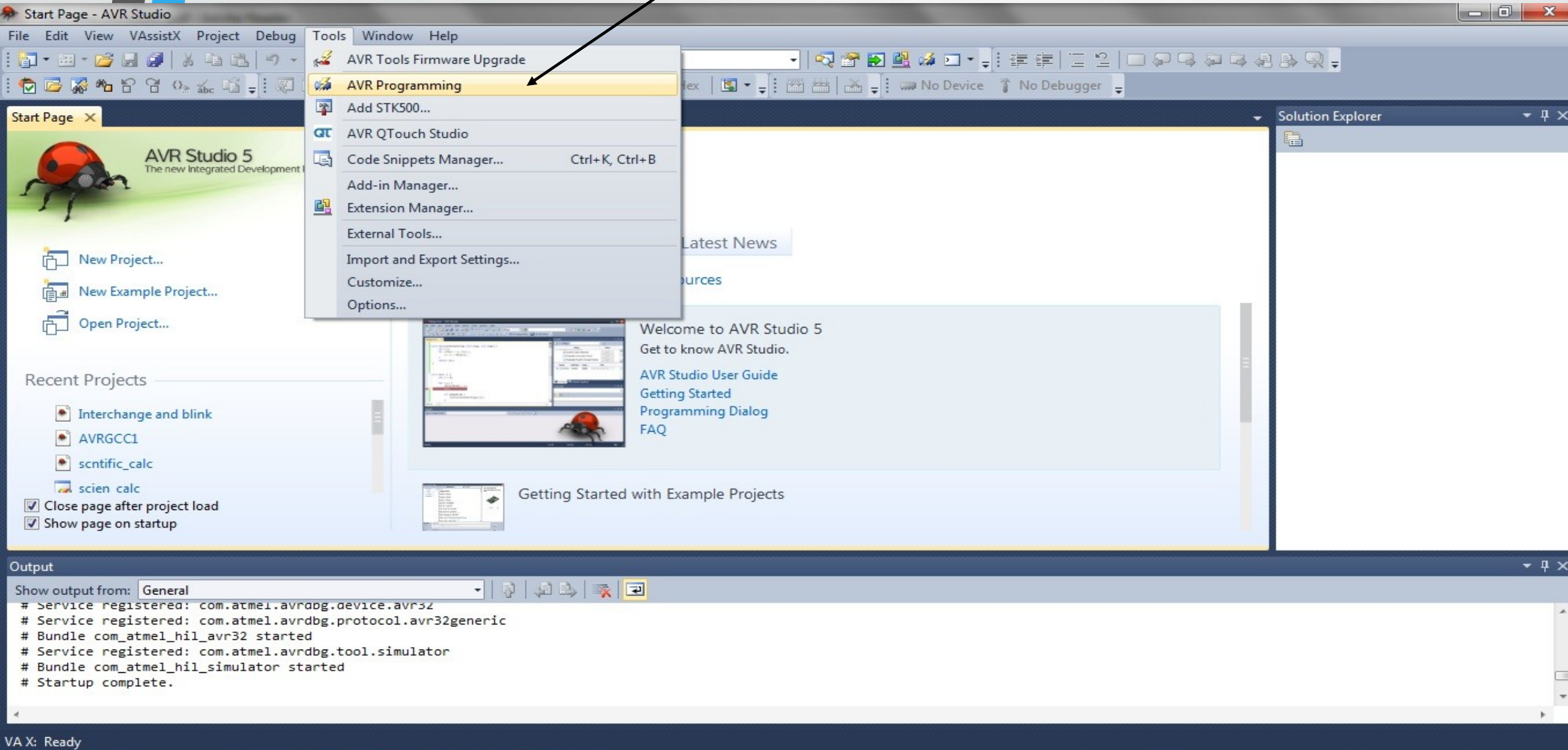
The screenshot shows the AVR Studio 5 Start Page. The 'Tools' menu is open, and the 'Add STK500...' option is highlighted with a yellow background and a black arrow pointing to it. The menu also includes options like 'AVR Tools Firmware Upgrade', 'AVR Programming', 'AVR QTouch Studio', 'Code Snippets Manager...', 'Add-in Manager...', 'Extension Manager...', 'External Tools...', 'Import and Export Settings...', 'Customize...', and 'Options...'. The Start Page features a 'Start Page' tab, a 'New Project...' button, a 'New Example Project...' button, an 'Open Project...' button, and a 'Recent Projects' list with items like 'Interchange and blink', 'AVRGCC1', 'scntific\_calc', and 'scien\_calc'. There are also checkboxes for 'Close page after project load' and 'Show page on startup'. The main area displays a 'Welcome to AVR Studio 5' message and links to 'AVR Studio User Guide', 'Getting Started', 'Programming Dialog', and 'FAQ'. The 'Output' window at the bottom shows the following text:

```
Show output from: General
# Service registered: com.atmel.avrdbg.device.avr32
# Service registered: com.atmel.avrdbg.protocol.avr32generic
# Bundle com_atmel_hil_avr32 started
# Service registered: com.atmel.avrdbg.tool.simulator
# Bundle com_atmel_hil_simulator started
# Startup complete.
```

The Windows taskbar at the bottom shows the system clock as 19:31 on 13-05-2013, along with various application icons.

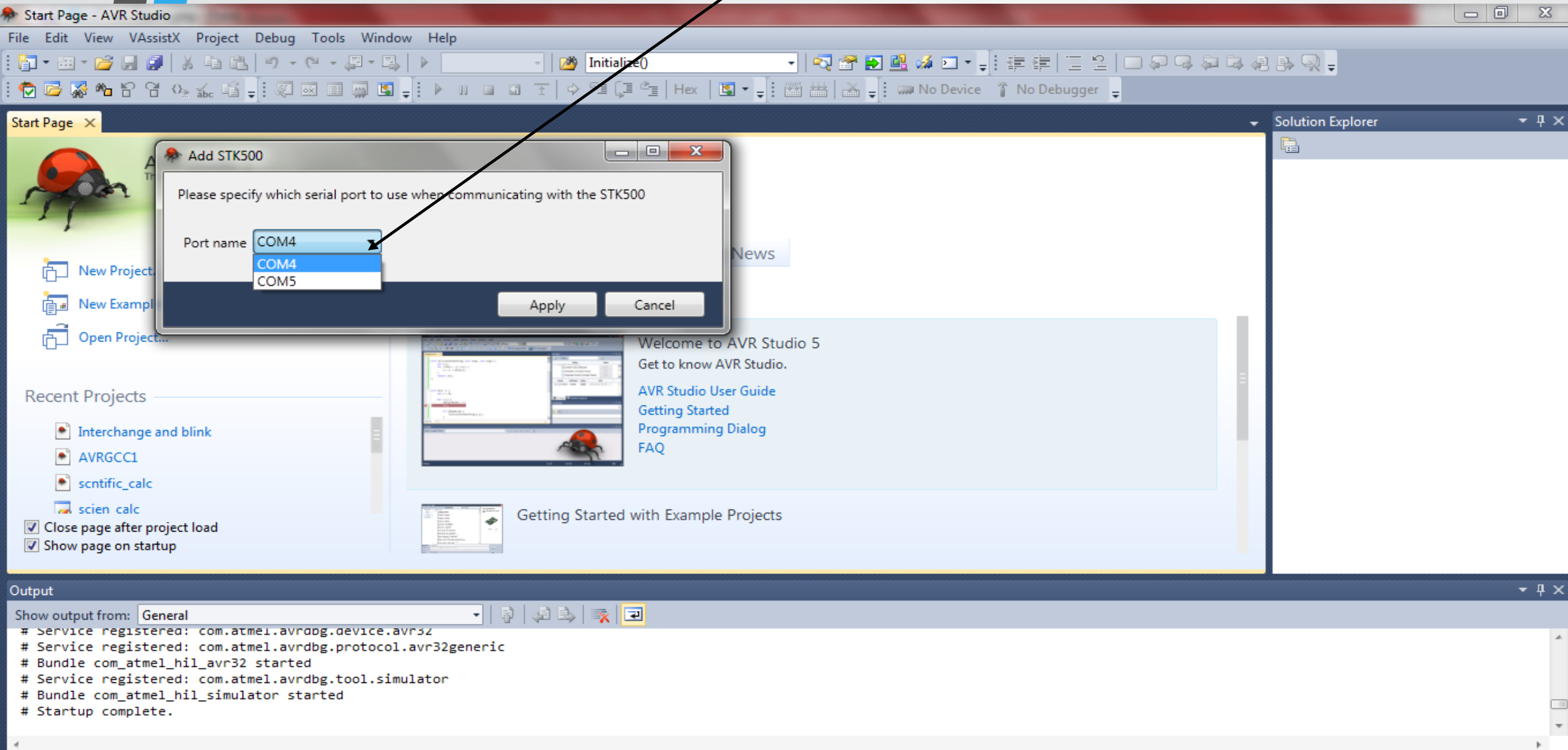


# Select AVR programming





# Select COM port



# Select Device -> Click Apply -> Read Device ID -> Read target Voltage -> Choose Hex File -> Then Program

The screenshot shows the AVR Studio 5 IDE with the AVR Programming dialog box open. The dialog box has several sections: Tool (STK500), Device (ATmega32), Interface (ISP), Device ID (not read), and Target Voltage (---). The 'Memories' section is expanded, showing the Flash memory with a file path: C:\Users\SHIVENDU's\scntific\_calc\default\scntific\_calc.hex. The 'EEPROM' section is also visible. The 'Program' button is highlighted. The 'Output' window at the bottom shows error messages: 'Getting clock value...Failed!', '# Failed to open COM3. Error 0x2.', and '[ERROR] TCF command: Tool:con'. The 'Solution Explorer' on the right shows a project structure.

AVR Studio 5  
The new Integrated Development Environment

File Edit View VAssistX Project Debug Tools Window Help

Start Page - AVR Studio

Interface settings  
Tool information  
Board settings  
Device information  
Memories  
Fuses  
Lock bits

Device  
Erase Device  
☒ Verify device after programming

Flash  
C:\Users\SHIVENDU's\scntific\_calc\default\scntific\_calc.hex  
☒ Erase device before programming  
Program Verify Read...

EEPROM  
Program Verify Read...

Getting clock value...Failed!

OK

Close

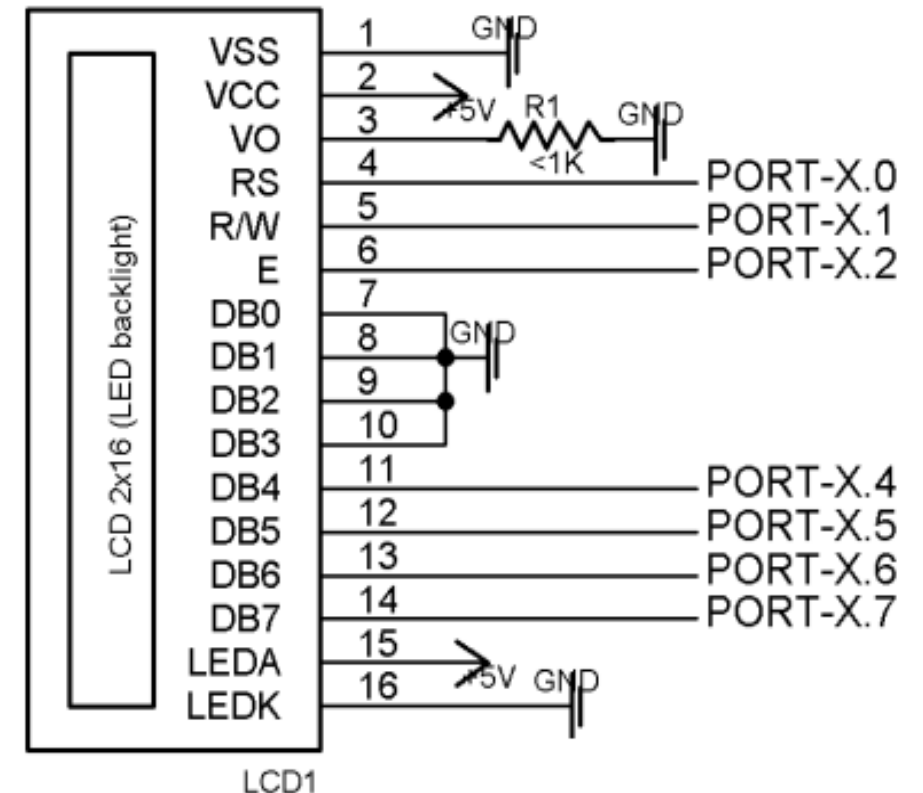
Output  
Show output from: General  
# Failed to open COM3. Error 0x2.  
19:49:48.452: [ERROR] TCF command: Tool:con  
# Failed to open COM3. Error 0x2.  
19:49:48.800: [ERROR] TCF command: Tool:con  
19:49:48.801: [ERROR] Unable to connect to  
# Failed to open COM3. Error 0x2.

Solution Explorer

19:51



- We interface an LCD to our microcontroller so that we can display messages, outputs, etc.
- Sometimes using an LCD becomes almost inevitable for debugging and calibrating the sensors
- We will use the 16x2 LCD, which means it has two rows of 16 characters each. Hence in total we can display 32 characters



# IR - TSOP Pair!





# Just Think Over!

- TSOP sensor detects the presence of light from the Infrared LED
- How will it distinguish from other Infrared light already present
- Should we use some kind of encoding ?
- TSOP sensor detects Infrared light only at 38 KHz
- How do we generate light at 38KHz?
- Timers and Interrupts... we will talk in the next lecture



# Thanks ㄹ



Shivendu Bhushan  
351 / Hall 2  
shivendu@iitk.ac.in  
8960419775



Sonu Agarwal  
368 / Hall 3  
sonuagr@iitk.ac.in  
9005888958



Swapnil Upadhyay  
F-102 / Hall 5  
swapnilu@iitk.ac.in  
7417279731, 7379599283

Website : <http://students.iitk.ac.in/eclub/index.php>

FB Group : <https://www.facebook.com/groups/eclub.iitk/>

E-mail : [eclub.iitk@gmail.com](mailto:eclub.iitk@gmail.com)

Youtube : <http://www.youtube.com/user/electronicclub>