Deeplang

面向IOT的程序设计语言

1. 项目背景介绍

2. Deeplang 设计目标

3. Deeplang 特性提案

4. Deeplang 工程实现

5. 未来的方向

背景介绍 http://deeplang.org/

DEEPLANG是由华为编程语言实验室与浙江大学竺可桢学院合作共建的语言设计与实现的开源课题。由华为实验室研究员指导，多所大学高低年级同学参与实践。

2020/4/29 项目组初始化成功

2020/5/07 第一次CCB会议

2020/10/29 hello world

…….

2020/12/* 第一次参与线下大会

Empowering everyone to build reliable and efficient software.

Combine object-oriented and functional programming in one concise, high-level language.

There should be one—and preferably only one—obvious way to do it

More easy and more safe.

安全区域

Rust的内存安全有着很高的使用门槛，能不能降低其上手难度？

```
extern "C" {
    fn abs(input: i32) -> i32;
}

fn main() {
    unsafe {
        println!("Absolute value of -3 according to C: {}", abs(-3));
    }
}
```

Expert

Domain developer

```
safe {
    fun abs(input: i32) -> i32;
}

fun main() -> () {
    puts(abs(-3).toString());
}
```

代数数据类型

trait

Sum type

Product type

一致的ADT声明方式

```
type Show {
  show: This -> string
}

type List [
  Nil,
  Cons(head: i32, tail: List)
]

Cons(1, Nil).head // get head

type Person (
  age: i32,
  name: string
) impl Show {
  fun show (this: This) -> string {
    return "xxx"
  }
}

// desugar
type Person [
    Person(age: i32, name: string)
] impl Show {
  fun show (this: This) -> string {
    return "xxx"
  }
}
```

静态结构化类型

类型的相等测试由类型真实的结构而不是名字来决定。

```
fun useShow(data: Show) -> string {
    return data.show();
}

type Foo (
) impl {
    fun show (this: This) -> string {
        return "xxx";
    }
}

let foo: Foo = new Foo ();
useShow (foo); // is ok
```

委托

将一个子字段FIELD的成员都委托到当前层级。

```
type Animal (
    weight: i32
)

type Person (
    age: i32,
    name: string,
    as animal: Animal  // delegate
)

let p: Person = new Person();
p.weight // p.a.weight
```

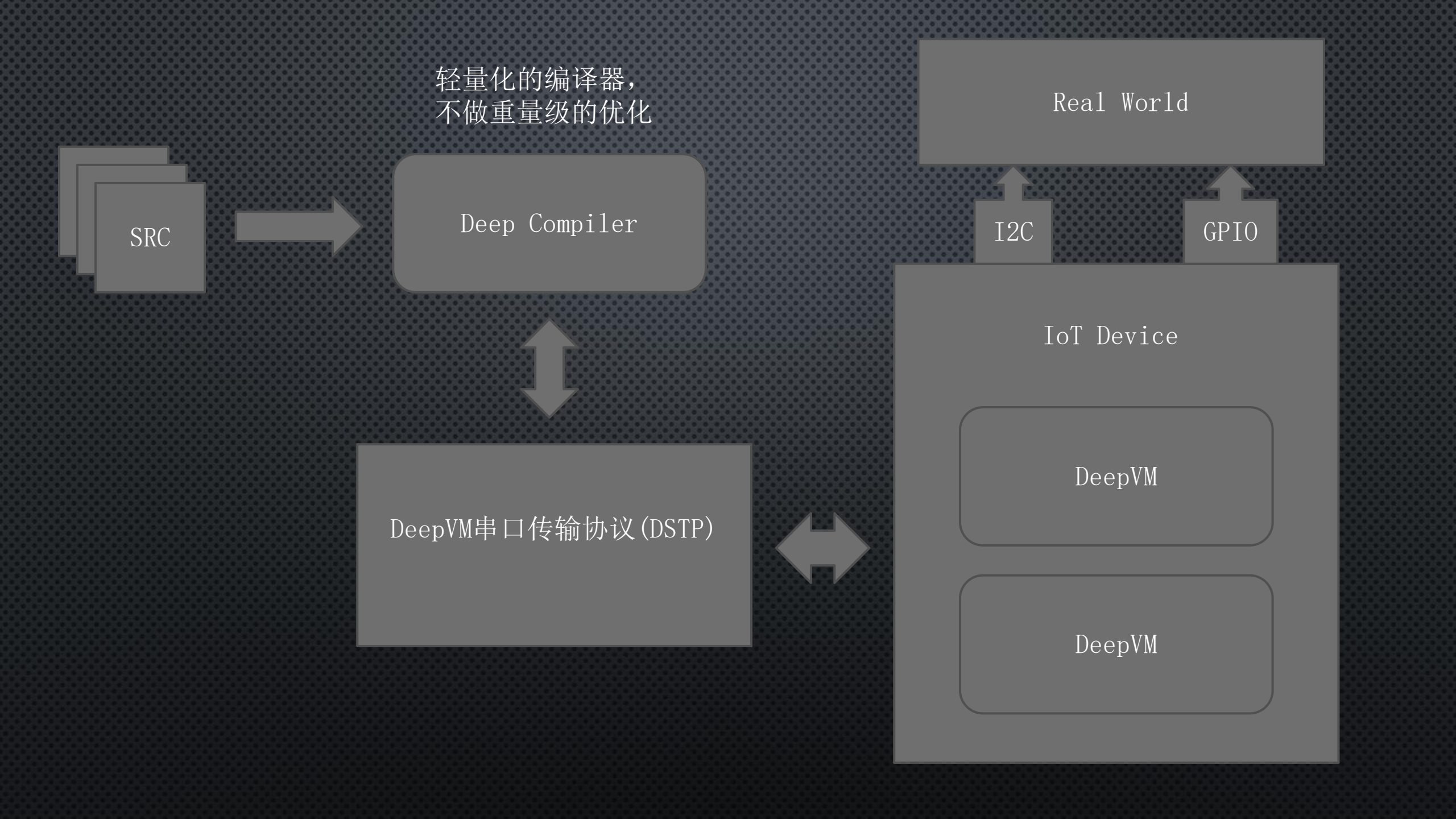
鸭子类型

采用组合的方式组织程序

不需要 impl Quack

满足结构即可通过 type checking

```
type Quack {
    quack: () -> ()
}

type Duck () impl Quack {
    fun quack () -> () {
        print("quaaaack");
    }
}

type Bird () {
    fun quack () -> () {
        print("bird quaaaack");
    }
}

fun sound(animal: Quack) -> () {
    animal.quack();
}

let duck: Duck = new Duck();
let bird: Bird = new Bird();

sound(duck); // quaaaack
sound(bird);  // bird quaaaack
```


轻量化的编译器，
不做重量级的优化
Real World
SRC
Deep Compiler
I2C
GPIO
IoT Device
DeepVM
DeepVM串口传输协议(DSTP)
DeepVM

未来的工作

1. 继续完善语言内核
2. 手写的更轻量级的lexer和parser
3. 替换wabt库
4. 新的后端，如riscv native

Deep Compiler

1. 针对Arm，RISCV平台优化后端运行效率
2. JIT

DeepVM

生态

1. 拥抱IoT平台，如鸿蒙，liteOS等。
2. 抽象硬件接口的标准库设计
3. 支持图形化Debug
4. Language server protocal
5. 包管理器